Javascript

Mit **Javascript** kannst du Szenen mit eigener Logik erweitern – z. B. UI anpassen, kleine Effekte einbauen, Zeiten messen oder externe Schnittstellen triggern.

> **Wichtig:** JS ist **inkonstant** – beim Betreten **jeder** Szene wird es **neu** ausgeführt. Teile, die überall gelten sollen, gehören in die **Masterszene** `[[ ]]`.
> Starte dein Script immer **leicht verzögert**, damit die Szene fertig gerendert ist:

```
setTimeout(function () { // dein Code hier }, 300);
```

Gute Praxis (sehr wichtig)

- **Idempotent denken:** Code darf mehrfach laufen, ohne doppelten Effekt zu erzeugen.
  Nutze einen „Once“-Guard:

  ```
  setTimeout(function () {
    if (window.__sceneOnce) return;
    window.__sceneOnce = true;
    // init …
  }, 200);
  ```
- **Aufräumen:** Wenn du `setInterval`/Event-Listener setzt, räume sie auf (oder nutze `{ once:true }` bei `addEventListener`).
- **Robust warten:** Wenn du ein Element brauchst, das evtl. später kommt, warte kurz darauf:

  ```
  function waitSel(sel, cb, tries=20){ const el = document.querySelector(sel); if (el) return cb(el); if (tries--) return setTimeout(()=>waitSel(sel, cb, tries), 100); } setTimeout(()=>waitSel('.btn-primary', btn=>{ btn.classList.add('pulse'); }), 200);
  ```
- **Keine Blocker:** Lange Schleifen vermeiden; lieber Timings/Promises verwenden.
- **Fallbacks:** Netzaufrufe können an CORS/Offline scheitern ? Fehler behandeln.

Häufige Mini-Rezepte

1) Button-Label/Style anpassen

```
setTimeout(function () {
  const btn = document.querySelector('button');
  if (!btn) return;
  btn.textContent = 'Los geht’s';
  btn.style.filter = 'drop-shadow(0 6px 12px rgba(0,0,0,.25))';
}, 200);
```

2) Szene nach X Sekunden automatisch fortsetzen

```
setTimeout(function () {
  // z.B. per Klick auf einen Weiter-Button simulieren
  const next = document.querySelector('[data-action="next"], .btn-continue');
  if (next) next.click();
}, 10000); // 10 s
```

3) Einfache „Einmal“-Animation

```
setTimeout(function () {
  const el = document.querySelector('.dialog, .content');
  if (!el) return;
  el.animate([{opacity:0, transform:'translateY(8px)'},{opacity:1, transform:'none'}],
             {duration:400, easing:'ease-out'});
}, 150);
```

4) Soft-Gate per Tageszeit (Beispiel)

```
setTimeout(function () {
  const h = new Date().getHours();
  if (h < 8 || h > 20) {
    alert('Tipp: Diese Aufgabe klappt am besten bei Tageslicht.');
  }
}, 250);
```

5) Ereignis mitzählen (lokal)

```
setTimeout(function () {
  const k = 'tries_safe_open';
  const n = 1 + +(localStorage.getItem(k) || 0);
  localStorage.setItem(k, n);
  if (n >= 3) {
    console.log('3 Fehlversuche – zeige Hilfe.');
    // hier könntest du z.B. einen Hilfe-Button einblenden
  }
}, 200);
```

Typische Stolperfallen

- **Doppelte Listener**: Bei jeder Rückkehr in die Szene wird JS neu ausgeführt ? `once`-Option nutzen:

  `window.addEventListener('click', handler, { once:true });`
- **Zu früher Zugriff**: Ohne Delay (`setTimeout`) existieren DOM-Elemente evtl. noch nicht.
- **Starke Manipulation am Layout**: Bedenke, dass die App auf vielen Displaygrößen läuft – nur sanft stylen.
- **Blockierende Prompts/Alerts**: Sparsam einsetzen; sie unterbrechen den Flow.

Checkliste

1. **Delay** einbauen (`setTimeout` 150–300 ms).
2. **Idempotent** & ggf. **Once-Guard** verwenden.
3. Wo nötig: **waitSel**/Observer nutzen.
4. Netz-/Fehlerfälle abfangen.
5. Auf mehreren Geräten testen.

Damit bekommst du stabile, kleine Scripts, die zuverlässig mit der Szenenlogik zusammenspielen.

Gute Beispiele "ready to use" findest du auch unter Hacks.

Revision #5
Created 27 September 2025 15:31:02 by David Wright
Updated 30 September 2025 11:52:02 by David Wright