# 018-Barrierefreie-Komponenten

Frontend-Aufgabe
Anforderungsniveau: Schwer

## 1 Ziel der Aufgabe

Sie bauen drei zusammengesetzte Bedienelemente von Grund auf nach den WAI-ARIA Authoring Practices: modaler Dialog, Auswahlfeld mit Filterung und Tab-Leiste. Fertige Komponentenbibliotheken sind ausgeschlossen – gerade das Verhalten, das solche Bibliotheken verstecken, ist der Prüfgegenstand. Bewertet werden Fokusverwaltung, Tastaturbedienung und die Ausgabe für Screenreader, nicht das Aussehen.

- React mit TypeScript (Vite oder Next.js), strikt typisiert, kein `any`
- WAI-ARIA-Rollen, -Zustände und -Eigenschaften von Hand gesetzt
- Fokusverwaltung: Fokusfalle, Fokusrückgabe, Roving Tabindex
- Tastaturbedienung mit Pfeiltasten, Home/End und Escape
- Eine Live-Region für eine Zustandsänderung ohne visuelles Gegenstück
- Tests mit Vitest und Testing Library, Prüfung mit axe-core

Veranschlagte Bearbeitungszeit: 5–7 Stunden

## 2 Aufgabenbeschreibung

1. **Projekt aufsetzen**

   Legen Sie ein React-Projekt mit TypeScript an (Vite oder Next.js), Styling nach Ihrer Wahl. **Nicht erlaubt** ist jede Bibliothek, die eines der drei Bedienelemente oder dessen Verhalten mitliefert – namentlich Radix UI, Headless UI, React Aria, Reach UI, MUI, Mantine, Ark UI, Ariakit, Chakra, Bootstrap, Downshift, cmdk, Floating UI und focus-trap. Fokusfalle und Tastaturlogik schreiben Sie selbst. Router, State-Bibliotheken, Icon-Sets und Testwerkzeuge sind erlaubt.

2. **Modaler Dialog**

   Ein Auslöser öffnet einen Dialog, der die Seite dahinter blockiert.

   - `role="dialog"`, `aria-modal=ttrue"`, Beschriftung über `aria-labelledby`, beschreibender Text über `aria-describedby`
   - Beim Öffnen wandert der Fokus in den Dialog, beim Schließen zurück auf den auslösenden Knopf
   - Tab und Shift+Tab bleiben im Dialog gefangen und laufen zyklisch um
   - Escape schließt, ein Klick auf die Abdunklung ebenfalls, ein Klick im Dialog nicht
   - Der Hintergrund ist für Screenreader unerreichbar (`inert` oder `aria-hidden` auf dem Seitencontainer, nicht auf dem Dialog)

- Das Scrollen der Seite wird gesperrt, ohne dass der Inhalt springt

3. **Auswahlfeld mit Filterung (Combobox)**

   Ein Textfeld filtert eine Liste von Einträgen. Verwenden Sie das Muster *Combobox mit Listbox-Popup*, nicht ein umgebautes `select`.

```
<label id="ort-label" for="ort">Standort</label>
<input
  id="ort"
  role="combobox"
  aria-expanded="true"
  aria-controls="ort-liste"
  aria-activedescendant="ort-opt-3"
  aria-autocomplete="list"
  autocomplete="off"
/>
<ul id="ort-liste" role="listbox" aria-labelledby="ort-label">
  <li id="ort-opt-3" role="option" aria-selected="true">Dresden</li>
</ul>
```

   Die Einträge stehen als feste Liste im Code, eine echte Datenquelle ist nicht nötig.

   - Der DOM-Fokus bleibt im Textfeld; die aktive Zeile wird nur über `aria-activedescendant` ausgezeichnet
   - Pfeil runter/hoch bewegen die Auswahl und rollen sie in den sichtbaren Bereich, Home/End springen an den Rand, Enter übernimmt, Escape schließt die Liste
   - Eine Live-Region (`aria-live=ppolite"`) meldet die Trefferzahl, entprellt, damit nicht jeder Tastenanschlag vorgelesen wird. Kein Treffer: sichtbare Meldung, die ebenfalls angesagt wird

4. **Tabs**

   Eine Tab-Leiste mit mindestens vier Reitern und je einem Panel.

   - `role=ttablist"`, `role=ttab"` mit `aria-selected` und `aria-controls`, `role=ttabpanel"` mit `aria-labelledby`
   - Roving Tabindex: genau ein Reiter ist über Tab erreichbar (`tabindex="0"`), alle anderen tragen `tabindex=1"`
   - Pfeil links/rechts wechseln zyklisch, Home/End springen an den Rand
   - Manuelle Aktivierung: der Fokus wandert mit den Pfeiltasten, das Panel wechselt erst bei Enter oder Leertaste
   - Enthält ein Panel keine fokussierbaren Elemente, bekommt es `tabindex="0"`, sonst nicht

5. **Demoseite**

   Eine Seite zeigt alle drei Bedienelemente mit je einem kurzen Beispiel, dazu eine nur bei Fokus sichtbare Sprungmarke „Zum Inhalt“.

6. **Tests**

   Schreiben Sie Tests mit Vitest, `@testing-library/react` und `@testing-library/user-event`. Fragen Sie Elemente über Rolle und zugänglichen Namen ab, nicht über CSS-Klassen oder Test-IDs. Mindestens abzudecken:

   - Dialog: Fokus landet nach dem Öffnen im Dialog, Tab läuft zyklisch um, Escape schließt, Fokus liegt danach wieder auf dem Auslöser

- Combobox: `aria-activedescendant` zeigt nach Pfeil runter auf die erste Option; Enter übernimmt den Wert; die Live-Region enthält die Trefferzahl
- Tabs: Pfeil rechts auf dem letzten Reiter landet auf dem ersten; das Panel bleibt bis Enter unverändert
- Ein Durchlauf mit `axe-core` über die Demoseite, im Ruhezustand und mit geöffnetem Dialog. Die Anbindung wählen Sie (etwa `jest-axe` oder `vitest-axe` unter Vitest)

## 3 Anforderungen

- React mit TypeScript, `strict` eingeschaltet, kein `any`. Scripts `dev`, `build`, `test`, `lint` in der `package.json`.
- Keine der unter Punkt 1 genannten Komponentenbibliotheken; die `package.json` muss das belegen.
- Icons ausschließlich aus **einem** Set – Lucide oder Phosphor, kein Mischen, keine Emojis als Bedienelement-Icons. Jedes Icon ist entweder dekorativ (`aria-hidden=ttrue"`) oder trägt einen zugänglichen Namen; beides zugleich gibt es nicht.
- Sichtbarer Fokusindikator auf allen bedienbaren Elementen, Kontrast mindestens 3:1. `outline: none` ohne Ersatz ist ein Ausschlusskriterium.
- Textkontrast nach WCAG 2.1 AA (4,5:1). Information niemals allein über Farbe. Animationen respektieren `prefers-reduced-motion`.
- Alles ist ohne Maus bedienbar. Es gibt keinen Zustand, aus dem die Tastatur nicht mehr herausfindet.
- Kein Formatierer ohne Projektkonfiguration: Prettier- bzw. ESLint-Konfiguration gehört mit ins Repository.

## 4 Abgabe

- Git-Repository mit nachvollziehbarer Historie, ohne `node_modules` und Build-Artefakte. Die Demoseite muss nach `npm install` und `npm run dev` ohne weitere Schritte erreichbar sein.
- Eine `README.md` mit:
  - Startanleitung (Installation, Entwicklungsserver, Tests, Lint), lauffähig mit Node 20 oder neuer
  - einer Tabelle je Bedienelement, welche Taste was auslöst
  - einer kurzen Notiz zu bewussten Abweichungen von den Authoring Practices samt Begründung

## 5 Bewertungskriterien

- **Fokusverwaltung** – Fokus geht nie verloren und landet nach jedem Öffnen und Schließen an der erwarteten Stelle, Roving Tabindex korrekt.
- **Tastaturbedienung** – vollständig nach den Authoring Practices, einschließlich Home/End und zyklischem Umlauf.

- **ARIA-Auszeichnung** – Rollen und Zustände passen zueinander und werden aktuell gehalten. Kein `aria-label` dort, wo ein sichtbares Label gehört; kein `aria-hidden` auf fokussierbaren Elementen.
- **Screenreader-Ausgabe** – Name, Rolle und Wert werden verständlich angesagt; die Live-Region meldet das Nötige und nicht mehr.
- **Testqualität** – Tests prüfen das Verhalten über Rollenabfragen und decken auch die leicht übersehenen Zustände ab (geöffneter Dialog, leeres Filterergebnis).
- **Code-Qualität** – klare Trennung zwischen Zustandslogik und Darstellung, wiederverwendbare Hooks, saubere Typen, lesbare Namen.
- **Einhaltung der Vorgaben** – ein Icon-Set, kein Layout-Sprung, sichtbarer Fokus überall.
- **Dokumentation** – die README ermöglicht es, die Abgabe in unter fünf Minuten zu starten und gezielt zu prüfen.

Viel Erfolg bei der Umsetzung.