017-Responsive-Navigation-Umbruchpunkte

Frontend-Aufgabe
Anforderungsniveau: Mittel

1 Ziel der Aufgabe

Sie bauen eine Navigationsleiste, die über drei Umbruchpunkte hinweg drei verschiedene Gestalten annimmt: volle Leiste, verkürzte Leiste, Menüschublade. Das ist die eine Hälfte der Aufgabe. Die andere, und für uns die wichtigere, ist der Nachweis: Sie sichern die Navigation mit automatisierten Tests ab und weisen nach, dass jeder Umbruchpunkt und jedes Bedienelement tatsächlich von mindestens einem Test erreicht wird.

Geprüft werden:

- Responsives Layout mit CSS Media Queries oder Container Queries
- Zustandsführung einer Navigation über Breitenwechsel hinweg
- Tastaturbedienung und ARIA-Attribute (`aria-expanded`, `aria-controls`, Fokusfalle in der Schublade)
- Ladezustand ohne Layout-Sprung
- Automatisierte Tests mit Playwright, inklusive der Frage, was ein grüner Test eigentlich belegt

Veranschlagte Bearbeitungszeit: 6–8 Stunden

2 Aufgabenbeschreibung

1. **Projekt aufsetzen**

   Legen Sie ein Next.js-Projekt mit TypeScript an (App Router) und richten Sie Playwright ein. Andere Frameworks sind zulässig, wenn Sie die Wahl in der README begründen; Playwright als Testwerkzeug ist gesetzt.

   ```
   npx create-next-app@latest navigation --typescript --app
   cd navigation
   npm init playwright@latest
   ```

2. **Die drei Umbruchpunkte**

   Die Navigation nimmt genau drei Gestalten an. Die Grenzen dürfen Sie verschieben, wenn Sie das in der README begründen – drei unterscheidbare Zustände bleiben Pflicht.

   - **Volle Leiste** ab 1200 px: Logo, alle Hauptpunkte mit Beschriftung, Untermenü als Aufklappfeld, Suchschaltfläche, Kontakt-Schaltfläche.
   - **Verkürzte Leiste** von 768 px bis 1199 px: Logo, die Hauptpunkte nur noch als Symbole ohne Beschriftung, die überzähligen Punkte wandern in ein Überlaufmenü („Mehr“). Welche Punkte überzählig sind, entscheiden Sie; halten Sie die Regel in der README fest. Die Suchschaltfläche bleibt als Symbolschaltfläche.

- **Menüschublade** unter 768 px: In der Leiste nur Logo und Menüschaltfläche. Alle Punkte liegen in einer seitlich einfahrenden Schublade, das Untermenü darin aufklappbar.

3. **Bedienelemente**

   Folgende Elemente müssen in mindestens einem Zustand vorhanden und bedienbar sein. Notieren Sie sich diese Liste – Sie brauchen sie in Schritt 7 wieder.

   - Fünf Hauptpunkte als Links, darunter „Leistungen“ mit drei Unterpunkten
   - Überlaufmenü „Mehr“ (nur verkürzte Leiste)
   - Menüschaltfläche zum Öffnen und Schließen der Schublade
   - Symbolschaltfläche Suche. Sie klappt ein Eingabefeld auf und setzt den Fokus hinein; eine Suchfunktion dahinter ist nicht Teil der Aufgabe.
   - Kontakt-Schaltfläche als hervorgehobene Aktion

   Untermenü und Schublade öffnen und schließen per Maus und per Tastatur, schließen bei `Escape` und bei Klick nach außen. Die Schublade hält den Fokus, solange sie offen ist, und gibt ihn beim Schließen an die auslösende Schaltfläche zurück.

4. **Menüpunkte werden geladen**

   Die Menüpunkte kommen nicht aus einer Konstanten im Bauteil, sondern von einem Route Handler im eigenen Projekt, der die Antwort um 600 ms verzögert:

```
// app/api/navigation/route.ts
import { NextResponse } from "next/server";

const punkte = [
  { id: "start",       label: "Start",       href: "/" },
  { id: "leistungen",  label: "Leistungen",  href: "/leistungen",
    kinder: [
      { id: "software", label: "Softwareentwicklung", href: "/leistungen/software"
    },
      { id: "hosting",  label: "Hosting",             href: "/leistungen/hosting" },
      { id: "security", label: "IT-Sicherheit",       href: "/leistungen/security"
    },
    ] },
  // ... drei weitere Punkte
];

export async function GET() {
  await new Promise((r) => setTimeout(r, 600));
  return NextResponse.json({ punkte });
}
```

   Bis die Antwort da ist, zeigt die Leiste einen Skeleton-Loader in der Zielgröße. Wenn die echten Punkte eintreffen, darf sich an der Höhe und an der Position der übrigen Elemente nichts verschieben.

5. **Verhalten beim Breitenwechsel**

   Ein Zustandswechsel darf die Navigation nicht in einen widersprüchlichen Zustand bringen. Prüfen Sie mindestens:

   - Schublade offen, Fenster wird auf 1000 px verbreitert: die Schublade verschwindet, ihr Inhalt ist über die verkürzte Leiste erreichbar, der Rumpf ist wieder rollbar.

- Überlaufmenü offen, Fenster wird auf 1400 px verbreitert: das Aufklappfeld schließt, die Punkte stehen ausgeschrieben in der Leiste.
- Zu keiner Breite sind zwei Navigationen gleichzeitig im Zugänglichkeitsbaum. Verstecken Sie den jeweils inaktiven Zweig nicht nur optisch.

6. **Automatisierte Absicherung**

   Übernehmen Sie die folgende Playwright-Konfiguration und den folgenden Rauchtest zunächst unverändert in Ihr Projekt. Beide sind Ausgangspunkt, nicht Ziel.

```
// playwright.config.ts (Auszug)
import { defineConfig } from "@playwright/test";

export default defineConfig({
  testDir: "./e2e",
  use: { baseURL: "http://localhost:3000" },
  projects: [
    { name: "desktop", use: { viewport: { width: 1440, height: 900 } } },
    { name: "mobile",  use: { viewport: { width: 390,  height: 844 } } },
  ],
});
```

```
// e2e/navigation.spec.ts (Vorgabe)
import { test, expect } from "@playwright/test";

test("alle Bedienelemente sind erreichbar", async ({ page }) => {
  await page.goto("/");
  const bedienelemente = page.locator("header button, header .btn");
  await expect(bedienelemente.first()).toBeVisible();
  for (const el of await bedienelemente.all()) {
    await expect(el).toBeEnabled();
  }
});
```

   Sobald die Leiste steht, läuft dieser Test grün durch. Nehmen Sie das nicht als Beleg, sondern als Behauptung, und prüfen Sie sie nach, bevor Sie darauf aufbauen. Ein Testlauf ist auch dann grün, wenn eine Prüfschleife leer durchläuft, wenn ein Element im DOM liegt, ohne sichtbar oder bedienbar zu sein, oder wenn ein Zustand der Oberfläche unter keinem der gefahrenen Viewports überhaupt entsteht. Verschaffen Sie sich deshalb zuerst Gewissheit darüber, welche der drei Gestalten dieser Lauf betritt und wie viele Elemente er dabei tatsächlich anfasst – etwa indem Sie sich die Trefferzahl des Locators je Projekt ausgeben lassen.

   Bauen Sie die Testabdeckung anschließend so aus, dass jeder der drei Zustände und jedes der in Schritt 3 genannten Bedienelemente von mindestens einem Test tatsächlich betreten beziehungsweise erfasst wird. Ergänzen Sie Tests für Tastaturbedienung, Fokusrückgabe, Skeleton-Zustand und die Breitenwechsel aus Schritt 5. Sie dürfen Konfiguration und Rauchtest dabei beliebig ändern oder ersetzen.

7. **Nachweis der Abdeckung**

   Legen Sie in der README eine Tabelle an: eine Zeile je Bedienelement, eine Spalte je Umbruchpunkt, in jeder Zelle der Name des Tests, der dieses Element in diesem Zustand anfasst, samt dem Playwright-Projekt, unter dem er läuft – oder ein begründetes „entfällt“.

   Bleibt beim Ausfüllen eine Zelle, Zeile oder Spalte leer, ist das ein Ergebnis und kein Versehen: halten Sie in zwei bis drei Sätzen fest, woran es lag und was Sie daraufhin geändert haben. Dasselbe gilt für jede Abweichung von der Vorgabe aus Schritt 6.

3 Anforderungen

- Drei unterscheidbare Zustände über drei Umbruchpunkte, ohne waagerechtes Rollen des Rumpfes bei 320 px Breite.
- Symbole ausschließlich aus **einem** Set – Lucide *oder* Phosphor, kein Mischen. Keine Emojis als Bediensymbole.
- Jede Symbolschaltfläche und jeder auf ein Symbol verkürzte Menüpunkt trägt einen Tooltip mit der vollen Beschriftung sowie ein `aria-label`. Tooltips nur dort, wo etwas abgekürzt oder nur als Symbol dargestellt ist.
- Für die ladenden Menüpunkte ein Skeleton-Loader in Zielgröße, kein Spinner. Kein Layout-Sprung beim Eintreffen der Daten.
- Tastaturbedienung vollständig: `Tab`, `Shift+Tab`, `Enter`, `Escape`. Sichtbarer Fokusring. Die Attribute `aria-expanded` und `aria-controls` sind korrekt gesetzt.
- Tests laufen mit `npx playwright test` ohne Handgriffe durch und sind reproduzierbar grün.
- Kein Formatierer ohne Projektkonfiguration. Wenn Sie Prettier oder ESLint einsetzen, legen Sie die Konfiguration im Repository ab.
- Keine echten Zugangsdaten, Kundenkennungen oder Schlüssel in Quellcode, Tests oder Konfiguration – auch nicht als Beispielwert.

4 Abgabe

- Ein Git-Repository mit nachvollziehbarer Commit-Historie. Kein einzelner „Initial commit“ mit dem fertigen Stand.
- Eine README mit: Startanleitung (Installation, Entwicklungsserver, Testlauf), gewählten Umbruchpunkten samt Begründung, der Abdeckungstabelle aus Schritt 7 und der Notiz zu den Änderungen an der Vorgabe.
- Die Playwright-Konfiguration und alle Testdateien liegen im Repository.
- Drei Bildschirmfotos, eines je Zustand, in der README eingebunden.

5 Bewertungskriterien

- **Testabdeckung (40 %):** Werden alle drei Zustände von Tests wirklich betreten? Erfassen die Prüfungen alle Bedienelemente oder nur die bequem greifbaren? Ist die Abdeckungstabelle ehrlich, also deckt sie sich mit dem, was die Tests tun?
- **Korrektheit der Navigation (25 %):** Verhalten an den Umbruchpunkten, Zustandswechsel, kein widersprüchlicher Zwischenzustand.
- **Zugänglichkeit (20 %):** Tastaturbedienung, Fokusführung, ARIA-Attribute, keine doppelte Navigation im Zugänglichkeitsbaum.
- **Umsetzung und Standards (15 %):** Ein Symbolset, Skeleton statt Spinner, Tooltips an den richtigen Stellen, lesbarer Code, verständliche README.

Positiv fällt auf, wer nicht nur Tests hinzufügt, sondern begründet, warum die vorhandenen nicht ausgereicht haben.