SOP 34

Navigation

Projektleitung | Geschäftsführung | IT | Recht

So baust du Leuchttürme für Screenreader-Nutzer:innen

Weil eine Website für blinde Menschen nicht visuell, sondern strukturell erfasst wird. Während sehende Nutzer:innen auf einen Blick zwischen Kopfzeile, Hauptinhalt und Footer unterscheiden, hören Screenreader-Nutzer:innen zunächst nur eine lange Abfolge von Texten und Links. Eine gute semantische Struktur ist wie eine Landkarte für diese Nutzer:innen. Sie teilt die Seite in klar benannte Regionen (sogenannte „Landmarks“) ein und ermöglicht es, mit einem Tastendruck direkt zum gewünschten Bereich zu springen – zum Beispiel von der Kopfzeile direkt zum Hauptinhalt, ohne durch die gesamte Navigation tabben zu müssen.

Wann musst du das machen?

- Bei der initialen Konzeption und Entwicklung des Grund-Templates deiner Website oder deines Themes.
- Wenn du die Haupt-Layoutbereiche einer Seite (Header, Footer, Sidebar) erstellst.
- Bei einem Barrierefreiheits-Audit, um grundlegende strukturelle Mängel zu beheben.

Welches Gesetz verlangt das?

- European Accessibility Act (EAA) / Barrierefreiheitsstärkungsgesetz (BFSG): Informationen und Beziehungen, die durch die Darstellung vermittelt werden, müssen programmatisch erkennbar sein.
- WCAG 2.1 , mehrere Kriterien sind hier von zentraler Bedeutung:
- 1.3.1 Info and Relationships: Das Kernkriterium. Die Struktur muss maschinenlesbar sein.
- 2.4.1 Bypass Blocks: Nutzer:innen müssen wiederkehrende Inhaltsblöcke (wie die Navigation) überspringen können. Landmarks sind der beste Weg, dies zu erfüllen.
- 4.1.2 Name, Role, Value: Landmarks definieren die „Rolle“ eines Seitenbereichs.

Der Prozess im Detail

1 🟧 **Phase 1: Das Grundgerüst – Die wichtigsten HTML5-Strukturelemente**

Moderne Websites sollten nicht nur aus -Containern bestehen. HTML5 bietet Elemente mit Bedeutung (Semantik), die automatisch Landmarks erzeugen.

- <header>: Für die Kopfzeile der Seite mit Logo und oft der Hauptnavigation.
- <nav>: Für jeden primären Navigationsblock.
- <main>: Für den einzigartigen Hauptinhalt der jeweiligen Seite. Wichtig: Es darf nur ein <main>-Element pro Seite geben.
- <footer>: Für die Fußzeile der Seite mit Impressum, Kontakt etc.
- <aside>: Für ergänzende Inhalte, die nicht direkt zum Hauptinhalt gehören (z. B. eine Sidebar mit weiterführenden Links).
- <section>: Gruppiert thematisch zusammengehörige Inhalte. Eine section sollte immer eine Überschrift haben.

2 🟨 Phase 2: ARIA-Landmarks – Wenn HTML allein nicht reicht

Die HTML5-Elemente aus Phase 1 sind die bevorzugte Methode. Manchmal ist man aber in einem System gefangen, das nur generische s erzeugt. In diesem Fall – und nur dann – kannst du mit ARIA-Rollen nachhelfen, um die s für Screenreader verständlich zu machen.

- (entspricht <header>)
- (entspricht <nav>)
- (entspricht <main>)
- (entspricht <footer>)
- (entspricht <aside>)
- : Dieses Attribut ist besonders nützlich für das Suchformular, da es kein direktes HTML5-Äquivalent gibt.

Grundregel: Sei nicht redundant. Wenn du das <nav>-Element nutzt, brauchst du nicht zusätzlich role="navigation".

3 🟩 Phase 3: Mehrere Navigationen klar benennen

Oft hat eine Seite mehrere Navigationsbereiche (Hauptmenü, Servicelinks im Footer, Kategorien in der Sidebar). Damit ein Screenreader-Nutzer sie unterscheiden kann, musst du ihnen einen einzigartigen Namen geben.

Nutze dafür das Attribut aria-label:

- <nav aria-label="Hauptnavigation">…</nav>
- <nav aria-label="Soziale Medien">…</nav>
- <nav aria-label="Rechtliche Informationen">…</nav>

So hört der Nutzer nicht dreimal „Navigation", sondern „Hauptnavigation", „Soziale Medien" und „Rechtliche Informationen".

4 🟥 Phase 4: %22Skip Links%22 – Die Überholspur zum Inhalt

Ein „Skip to main content"-Link ist die direkteste Umsetzung von WCAG 2.4.1. Er ist das allererste Element, das beim Drücken der Tab-Taste auf einer Seite erscheint.

- Funktionsweise: Der Link ist anfangs oft visuell versteckt. Er wird erst sichtbar, wenn er den Tastaturfokus erhält.
- Ziel: Ein Klick auf diesen Link überspringt den gesamten Header sowie die Navigation und setzt den Fokus direkt an den Anfang des <main>-Inhaltsbereichs.
- Umsetzung: Zum Hauptinhalt springen und im Hauptinhalt dann <main id="main-content">.

5 🟦 Phase 5: Die Struktur mit einem Screenreader prüfen

Der beste Weg zur Kontrolle ist, die Seite wie ein blinder Mensch zu erleben.

- Aktiviere einen Screenreader (NVDA für Windows, VoiceOver für macOS).
- Nutze das „Rotor"-Menü (bei VoiceOver) oder die Elementenliste (bei NVDA), um dir alle „Landmarks" oder „Regionen" anzeigen zu lassen.
- Prüfe: Werden alle wichtigen Bereiche (banner, navigation, main, contentinfo etc.) korrekt aufgelistet? Sind die Navigationen klar benannt? Komme ich mit einem Klick zur richtigen Stelle?

Ergebnis

Am Ende dieser SOP hast du:

- Eine Website, die für Screenreader-Nutzer:innen keine undurchdringliche Wand aus Inhalt mehr ist, sondern eine gut organisierte Landkarte.
- Die Navigationseffizienz für Menschen, die Hilfstechnologien nutzen, massiv verbessert.
- Fundamentale WCAG-Prinzipien zur Struktur und Bedienbarkeit zuverlässig erfüllt.
- Eine technisch saubere und professionelle Grundlage für alle deine Inhalte geschaffen.

Um die in dieser SOP festgelegten Anforderungen an Fokus-Management, Tastaturfalle, Schließ-Mechanismen und ARIA-Attribute für modale Dialoge und Overlays sofort und zuverlässig umzusetzen, ist es sinnvoll ein bereits erprobten Muster zu nutzen.

Das folgende Beispiel eines „Accessible Modal Dialog“ des W3C (World Wide Web Consortium) ist ein Industriestandard und erfüllt alle relevanten Kriterien der WCAG und des BFSG. Es dient als Direktive für die technische Implementierung .

Anforderung: Semantische HTML-Struktur für alle Seitentemplates

Hallo Team,

für alle neuen und bestehenden Seitentemplates ist eine saubere semantische Grundstruktur gemäß WCAG 1.3.1 verpflichtend.

Bitte stellt sicher, dass alle Templates wie folgt aufgebaut sind:

- HTML5-Landmarks verwenden: Die Hauptbereiche der Seite müssen mit <header>, <nav>, <main> und <footer> ausgezeichnet werden.
- Keine Redundanz: ARIA-Rollen (z. B. role=“main“) sind nur zu verwenden, wenn die semantischen HTML5-Elemente nicht genutzt werden können.
- Navigationen benennen: Wenn mehrere <nav>-Elemente auf einer Seite existieren, muss jedes ein eindeutiges aria-label erhalten (z. B. „Hauptnavigation“).
- Skip-Link implementieren: Jede Seite muss einen „Zum Hauptinhalt springen“-Link als erstes fokussierbares Element enthalten.

Diese Struktur ist die Grundlage für eine barrierefreie Navigation und nicht verhandelbar.

Danke!

Viele Grüße [Dein Name]

Fragen & Antworten

Warum ist es so wichtig, HTML5-Elemente (</p> <header>, <nav>, <main> usw.) anstelle von s mit ARIA-Rollen zu verwenden, auch wenn das Ergebnis für Screenreader dasselbe ist?

Das ist die Frage nach der progressiven Verbesserung (Progressive Enhancement), du! HTML5-Elemente sind nativ semantisch . Das bedeutet, Browser und assistive Technologien (wie Screenreader) verstehen ihre Bedeutung „out of the box“, ohne dass zusätzliche Attribute interpretiert werden müssen. Das macht den Code robuster, zukunftssicherer und oft auch einfacher zu warten . Wenn du s mit ARIA-Rollen verwendest, verlässt du dich darauf, dass ARIA korrekt interpretiert wird. Sollte ARIA aus irgendeinem Grund nicht geladen oder unterstützt werden, verliert dein seine Bedeutung, während ein <nav> -Element immer eine Navigation bleibt. Es ist die „natürlichere“ und damit die stabilere Lösung.

Ich habe ein CMS, das viele </p> s generiert und mir wenig Kontrolle über die HTML-Struktur gibt. Wie kann ich hier am besten vorgehen, um die SOP einzuhalten?

Das ist ein häufiges Problem, du. In solchen Fällen ist es entscheidend, die Grenzen deines CMS zu verstehen und clevere Workarounds zu finden:

- Nutze Themes/Plugins, die semantisches HTML verwenden: Prüfe, ob es Themes oder Plugins gibt, die von Haus aus HTML5-Elemente korrekt einsetzen.
- Greife mit JavaScript ein (als letzte Instanz): Wenn das CMS nur s ausspuckt, kannst du nach dem Laden der Seite mit JavaScript diese s erkennen und ihnen die entsprechenden role -Attribute hinzufügen. Dies ist allerdings ein Notnagel und sollte nur eingesetzt werden, wenn du die HTML-Ausgabe des CMS absolut nicht direkt beeinflussen kannst, da es eine clientseitige Abhängigkeit schafft.
- Nutze das CMS bewusst: Auch wenn das Grundgerüst aus s besteht, kannst du oft innerhalb von Inhaltsbereichen (<section> , <article>) die Struktur mit HTML5-Elementen verbessern.
- Spreche mit dem Anbieter: Leite die Anforderungen an deinen CMS-Anbieter weiter und fordere Verbesserungen bei der semantischen Ausgabe.

Ich habe verstanden, dass es nur ein <main>-Element pro Seite geben darf. Aber was ist, wenn ich mehrere Hauptinhaltsbereiche habe, die logisch eigenständig sind (z.B. bei einer Landingpage mit mehreren großen Abschnitten)?

Das ist ein klassisches Missverständnis der <main> -Rolle, du. Das <main> -Element sollte den einzigartigen Inhalt der aktuellen Seite umfassen. Wenn du auf einer Seite mehrere große, thematisch eigenständige Abschnitte hast, gruppierst du diese innerhalb des <main> -Elements mit <section> -Elementen. Jede <section> sollte dann eine eigene Überschrift (<h2> oder tiefer) haben, um die Struktur klar zu gliedern. Das <main> ist der Container für all diese Hauptinhalte, auch wenn sie aus mehreren logischen „Teilen“ bestehen. Es hilft, den Screenreader-Nutzern einen einzigen zentralen Ankerpunkt für den Kern der Seite zu bieten.

Meine Seite hat ein Suchfeld im Header und ein weiteres, detaillierteres Suchformular auf einer separaten Suchseite. Wie benenne ich diese role=%22search%22-Elemente korrekt, wenn ich zwei habe?

Ähnlich wie bei multiplen <nav> -Elementen musst du auch hier mit aria-label arbeiten, du. Auch wenn die role="search" -Attribute an sich schon von Screenreadern erkannt werden, gibst du ihnen für die Unterscheidung eindeutige Namen:

- ... (für das Suchfeld im Header)
- ... (für das detailliertere Formular auf der Suchseite) So weiß der Nutzer genau, welches Suchfeld er gerade bedient und wofür es gedacht ist, wenn er sich durch die Landmarks navigiert.

Wir haben dynamische Inhalte, die sich auf der Seite ändern (z.B. Ajax-geladene Inhalte oder Tabs, die neue Inhalte anzeigen). Bleiben die Landmarks und die Struktur bei solchen Änderungen erhalten oder muss ich etwas Besonderes beachten?

Das ist ein kritischer Punkt bei Single Page Applications (SPAs) oder Seiten mit viel dynamischem Laden, du!

- Stabile Landmarks: Die Haupt-Landmarks (<header> , <nav> , <main> , <footer>) sollten idealerweise statisch und immer präsent bleiben, auch wenn sich der Inhalt im <main> -Bereich ändert.
- Dynamische Inhalte im <main> : Wenn du neue Inhalte in den <main> -Bereich lädst (z.B. beim Wechsel eines Tabs), solltest du sicherstellen, dass die Überschriftenhierarchie innerhalb des <main> neu gesetzt oder aktualisiert wird und die neuen Inhalte semantisch korrekt (z.B. in <section> -Elementen mit Überschriften) strukturiert sind.
- Fokus-Management: Nach dem Laden neuer Inhalte ist es oft sinnvoll, den Fokus programmatisch auf die neue Hauptüberschrift des geladenen Inhalts zu setzen. Das signalisiert dem Screenreader-Nutzer, dass sich der Inhalt geändert hat und wo er mit dem Lesen beginnen soll.
- ARIA Live Regions: Für Statusmeldungen bei dynamischem Laden (z.B. „Inhalte werden geladen…“) solltest du aria-live="polite" oder role="status" nutzen, wie in deiner anderen SOP erwähnt.

Das Zusammenspiel von stabiler Grundstruktur und dynamischer, aber semantisch korrekter Befüllung ist hier der Schlüssel.