SOP 31

Slider, Karussells und Akkordeons

Projektleitung | Geschäftsführung | IT | Recht

So werden aus interaktiven Hürden nützliche Werkzeuge

Weil diese Komponenten oft die größten Barrierefreiheits-Fallen auf einer Website sind. Sie verstecken Inhalte, sind häufig nicht mit der Tastatur bedienbar und kündigen für Screenreader-Nutzer:innen nicht an, was gerade passiert. Ein automatisch abspielendes Karussell kann zudem extrem ablenken und verstößt ohne eine Pause-Funktion direkt gegen die WCAG. Wenn diese Elemente nicht sorgfältig umgesetzt werden, schließen sie Menschen aktiv aus, anstatt ihnen zu helfen.

Wann musst du das machen?

- Immer, wenn du planst, ein Akkordeon (z. B. für FAQs), ein Karussell (für Produkte, Testimonials) oder einen Slider einzusetzen.
- Bei einem Audit oder Relaunch von Seiten, die diese Elemente enthalten.
- Bevor du ein Plugin oder eine Bibliothek für diese Funktionalität kaufst oder installierst.

Welches Gesetz verlangt das?

- European Accessibility Act (EAA) / Barrierefreiheitsstärkungsgesetz (BFSG): Alle interaktiven Funktionen einer Dienstleistung müssen bedienbar sein.
- WCAG 2.1 , eine ganze Reihe von Kriterien ist hier relevant:
- 2.1.1 Keyboard: Alle Funktionen müssen per Tastatur erreichbar sein.
- 2.2.2 Pause, Stop, Hide: Für alle sich bewegenden, blinkenden oder automatisch aktualisierenden Inhalte muss es eine Steuerungsmöglichkeit geben.
- 4.1.2 Name, Role, Value: Damit Hilfstechnologien verstehen können, was ein Element ist, welche Rolle es hat und welchen Zustand es hat (z. B. „Schaltfläche, eingeklappt“).
- 2.4.3 Focus Order: Der Fokus muss sich in einer logischen Reihenfolge bewegen.
- 2.4.7 Focus Visible: Das aktive Bedienelement muss klar hervorgehoben sein.

Der Prozess im Detail

1. 🟧 **Phase 1: Die korrekte HTML-Struktur schaffen**

   Die Grundlage für alles Weitere.

   - Der Auslöser (Trigger): Nutze eine Überschrift (z. B. <h2> oder <h3>) für die Frage oder den Titel. In dieser Überschrift platzierst du einen <button>. Nur ein Button kann von Screenreadern korrekt als auslösbares Element erkannt werden.
   - Der Inhaltsbereich (Panel): Der Inhalt, der ein- und ausgeklappt wird, kommt in ein .

2 🟨 Phase 2: Zustände mit ARIA-Attributen kommunizieren

So wissen Screenreader, was gerade passiert.

- Auf dem <button>:
- aria-expanded="false": Gibt an, dass das Panel eingeklappt ist. Der Wert ändert sich per JavaScript zu „true", wenn es ausgeklappt wird.
- aria-controls="panel-id": Verknüpft den Button mit dem Inhalts-Panel. Das des Panels braucht dafür eine entsprechende id="panel-id".
- Auf dem Inhalts- :
- role="region" und aria-labelledby="button-id": Definiert das Panel als eine zusammengehörige Region, deren Titel vom Button (<button id="button-id">) geliefert wird.

3 🟩 Phase 3: Tastaturbedienung sicherstellen

- Die Tasten Enter und Leertaste müssen das jeweilige Panel öffnen und schließen, wenn der Fokus auf dem Button ist.
- Die Tab-Taste muss den Fokus logisch vom einen zum nächsten Akkordeon-Header bewegen.

4 🟥 Phase 4: Die wichtigste Regel – Kontrolle über Bewegung

- Pause-Button ist Pflicht: Wenn das Karussell automatisch rotiert, musst du einen gut sichtbaren und tastaturbedienbaren Button anbieten, um die Animation zu pausieren oder zu stoppen (WCAG 2.2.2).
- Keine Animation bei Hover: Die Rotation sollte auch anhalten, wenn sich der Mauszeiger oder der Tastaturfokus innerhalb des Karussells befindet.

5 🟦 Phase 5: Bedienelemente barrierefrei gestalten

- „Weiter"- und „Zurück"-Pfeile: Müssen echte <button>-Elemente sein. Gib ihnen einen zugänglichen Namen, der den Kontext erklärt, z. B. <button aria-label="Nächstes Produkt anzeigen">. Ein einfaches >-Symbol reicht nicht.
- Punkt-Navigation (Dots): Auch die Punkte unter dem Slider müssen eine Liste von <button>-Elementen sein. Ihr zugänglicher Name sollte das Ziel beschreiben, z. B. <button aria-label="Gehe zu Produkt 3″>.
- Sichtbarer Fokus: Alle diese Bedienelemente brauchen einen klaren Fokus-Indikator (siehe SOP 25).

6 🟪 Phase 6: Inhalte und Zustände für Screenreader kommunizieren

- Ankündigung bei Wechsel: Nutze das Attribut aria-live="polite" auf dem Bereich, der die Slides enthält. So wird ein automatischer Wechsel vom Screenreader angekündigt (z. B. „Slide 2 von 5 wird angezeigt").
- Inhalte erreichbar machen: Alle klickbaren Elemente innerhalb einer Slide (z. B. ein „Mehr erfahren"-Link) müssen mit der Tab-Taste erreichbar sein.
- Fokus-Management: Wenn ein Nutzer auf „Weiter" klickt, darf der Fokus nicht an den Anfang der Seite springen. Er sollte logisch auf dem „Weiter"-Button bleiben oder auf die neue Slide bewegt werden.

Ergebnis

Am Ende dieser SOP hast du:

- Komplexe interaktive Komponenten, die keine Barrieren mehr darstellen, sondern einen echten Mehrwert bieten.
- Wichtige und fortgeschrittene WCAG-Anforderungen erfüllt und deine Website rechtssicherer gemacht.
- Eine deutlich bessere Nutzererfahrung für alle geschaffen, indem du Frust vermeidest und Kontrolle ermöglichst.
- Konkrete, wiederverwendbare Anleitungen für deine Entwickler:innen.

Dieses Template dient dir als direkte Arbeitsanweisung für Entwickler:innen und Designer:innen, um Slider, Karussells und Akkordeons gemäß SOP 31 barrierefrei umzusetzen. Es fasst die technischen Anforderungen prägnant zusammen und dient als Checkliste für die Implementierung.

Template: Barrierefreie Implementierung von Slidern/Karussells/Akkordeons

Komponenten-Typ: [ ] Slider / [ ] Karussell / [ ] Akkordeon Bezeichnung der Komponente: [Hier den Namen eintragen, z.B. „FAQ-Akkordeon“, „Hero-Slider“, „Produktkarussell“]

Verantwortliche(r) Entwicklung: [Name des/der Entwickler:in/Teams] Verantwortliche(r) QA/Testing: [Name des/der Tester:in] Datum der Anfrage/Start: [Datum] Geplantes Go-Live: [Datum]

1. Korrekte HTML-Struktur (Umsetzung durch Entwicklung)

- Auslöser (Trigger):
- Ist eine Überschrift (<h2> oder <h3>) für Titel/Frage verwendet?
- Ja
- Ist ein <button> Element in der Überschrift platziert?
- Ja
- Inhaltsbereich (Panel):
- Liegt der Inhalt in einem Element?
- Ja

2. Zustände mit ARIA-Attributen kommunizieren (Umsetzung durch Entwicklung)

- Auf dem <button> :
- aria-expanded="false" (initial) / true (wenn ausgeklappt) dynamisch gesetzt?
- Ja
- aria-controls="panel-id" korrekt mit Inhalts- verknüpft?
- Ja
- Auf dem Inhalts- (Panel):
- role="region" gesetzt?
- Ja
- aria-labelledby="button-id" korrekt mit Button-ID verknüpft?
- Ja

3. Tastaturbedienung sicherstellen (Umsetzung durch Entwicklung / Prüfung durch QA)

- Enter und Leertaste öffnen/schließen Panel bei Fokus auf Button?
- Ja
- Nein, Problem bei [Beschreibung]
- Tab-Taste bewegt Fokus logisch (von Header zu Header)?
- Ja
- Nein, Fokus springt/ist unlogisch bei [Beschreibung]

4. Kontrolle über Bewegung (NUR bei Slidern/Karussells mit Auto-Rotation – Umsetzung durch Entwicklung / Prüfung durch QA)

- Pause-Button vorhanden, sichtbar und tastaturbedienbar (WCAG 2.2.2)?
- Ja
- N/A (keine Auto-Rotation)
- Nein, muss implementiert werden.
- Animation stoppt bei Hover (Mauszeiger/Tastaturfokus) innerhalb des Elements?
- Ja
- N/A (keine Auto-Rotation)
- Nein, Animation läuft weiter.

5. Bedienelemente barrierefrei gestalten (Umsetzung durch Entwicklung / Prüfung durch QA)

- „Weiter“-/“Zurück“-Pfeile:
- Sind echte <button> -Elemente?
- Ja
- Haben sie einen zugänglichen aria-label (z.B. „Nächstes Produkt anzeigen“)?
- Ja
- Nein, aria-label fehlt/ist unklar.
- Punkt-Navigation (Dots):
- Sind eine Liste von <button> -Elementen?
- Ja
- Haben sie einen zugänglichen aria-label (z.B. „Gehe zu Slide 3“)?
- Ja
- Nein, aria-label fehlt/ist unklar.
- Sichtbarer Fokus-Indikator:
- Alle Bedienelemente haben einen klaren, sichtbaren Fokus-Indikator?
- Ja
- Nein, Fokus ist nicht sichtbar bei [Element]

6. Inhalte und Zustände für Screenreader kommunizieren (Umsetzung durch Entwicklung / Prüfung durch QA)

- Ankündigung bei Wechsel (aria-live="polite") auf dem Slide-Container?
- Ja
- N/A (Akkordeon)
- Nein, muss implementiert werden.
- Alle klickbaren Elemente innerhalb einer Slide mit Tab-Taste erreichbar?
- Ja
- Nein, Elemente in Slide [Nummer] nicht erreichbar.
- Fokus-Management logisch (Fokus bleibt auf Button/bewegt sich zur neuen Slide)?
- Ja
- Nein, Fokus springt/ist unlogisch.

Zusätzliche Kommentare/Aktionspunkte: [Hier können spezifische Notizen, Herausforderungen oder besondere Umstände festgehalten werden.]

Fragen & Antworten

Warum ist es bei einem automatisch abspielenden Karussell so wichtig, eine Pause-Funktion anzubieten? Reicht es nicht, wenn es langsam rotiert?

Nein, eine langsame Rotation reicht nicht aus. Für Menschen mit bestimmten kognitiven Einschränkungen, Aufmerksamkeitsdefiziten oder Lernschwierigkeiten kann ein sich ständig bewegendes Element extrem ablenkend sein und das Lesen anderer Inhalte auf der Seite unmöglich machen. Es ist so, als würde jemand ständig vor dir herumlaufen, während du versuchst, ein Buch zu lesen. Auch Screenreader-Nutzer:innen können Schwierigkeiten haben, den Inhalt zu erfassen, wenn sich der Fokus ständig verschiebt. Eine Pause-Funktion gibt den Nutzern die Kontrolle zurück und ermöglicht es ihnen, die Inhalte in ihrem eigenen Tempo zu konsumieren oder die Bewegung ganz zu stoppen, wenn sie stört.

Ich verwende doch schon Pfeile für %22Weiter%22 und %22Zurück%22. Warum reicht ein einfaches >-Symbol im HTML nicht aus, und was ist mit Icons?

Ein einfaches > -Symbol oder ein reines Icon (ohne zusätzliche Beschreibung) ist für sehende Nutzer:innen oft intuitiv. Für Screenreader-Nutzer:innen ist es aber bedeutungslos , wenn es nicht korrekt ausgezeichnet ist. Der Screenreader würde nur „>“ oder „Bild“ vorlesen, was keinerlei Kontext liefert. Deshalb müssen die Pfeile echte <button> -Elemente sein und einen zugänglichen Namen (via aria-label) bekommen, der ihre Funktion beschreibt. Wenn du ein Icon verwendest, ist der aria-label noch wichtiger, da er die visuelle Information in Text umwandelt (z.B. <button aria-label="Nächste Folie">...</button>). So wissen Screenreader-Nutzer:innen genau, welche Aktion der Button auslöst.

Meine Akkordeons sind sehr lang und enthalten viel Text. Sollte ich dann trotzdem alles auf einmal ein- und ausklappen können, oder gibt es hier bessere Ansätze?

Bei sehr langen Akkordeons kann es sinnvoll sein, über eine Suchfunktion innerhalb des Akkordeons oder eine „Alles öffnen/schließen“-Funktion nachzudenken. Wenn ein Akkordeon Dutzende von Einträgen hat, wird es mühsam, jeden einzelnen anzuklicken. Eine „Alles öffnen/schließen“-Option kann Nutzern mit motorischen Einschränkungen oder Screenreader-Nutzern viel Zeit und Klicks ersparen. Wichtig ist, dass diese Funktionen selbst barrierefrei umgesetzt sind und klar kommunizieren, was sie tun. Achte auch darauf, dass der Fokus beim Öffnen oder Schließen von Akkordeons logisch innerhalb des Inhaltsbereichs bleibt.

Was ist der Unterschied zwischen aria-live=%22polite%22 und aria-live=%22assertive%22 für die Ankündigung von Inhaltswechseln, und wann sollte ich welches verwenden?

Beide aria-live -Attribute weisen Screenreader an, Änderungen in einem bestimmten Bereich der Webseite anzukündigen, aber sie tun dies auf unterschiedliche Weise:

- aria-live="polite" : Das ist die sanftere Variante. Der Screenreader unterbricht seine aktuelle Sprachausgabe nicht , um die Änderung anzukündigen. Er wartet, bis er mit dem aktuellen Satz oder Element fertig ist, bevor er die neue Information vorliest. Das ist ideal für nicht-kritische Updates, wie den Wechsel einer Slide in einem Karussell oder das Erscheinen einer Erfolgsmeldung, die keine sofortige Reaktion erfordert.
- aria-live="assertive" : Dies ist die unterbrechende Variante. Der Screenreader unterbricht sofort alles, was er gerade vorliest, um die neue Information anzukündigen. Das sollte nur für dringende, kritische und potenziell störende Nachrichten verwendet werden, die eine sofortige Aufmerksamkeit erfordern, wie eine Fehlermeldung, die den Nutzer zum Handeln zwingt (z.B. „Passwort falsch“).

Für Karussells und Slider ist aria-live="polite" fast immer die richtige Wahl, da der Wechsel der Folie zwar wichtig ist, aber in der Regel nicht sofort die gesamte Aufmerksamkeit des Nutzers erfordert und die laufende Interaktion nicht unterbrechen sollte.

-

Meine Inhalte in den Slidern sind nicht nur Text, sondern enthalten auch Bilder und Links. Müssen diese auch barrierefrei sein, wenn sie in einem Slider liegen?

Ja, absolut! Die Barrierefreiheit eines Sliders oder Karussells betrifft nicht nur die Steuerelemente und die Navigation, sondern alle Inhalte, die darin angezeigt werden . Das bedeutet:

- Bilder: Alle Bilder müssen aussagekräftige alt -Attribute haben, die ihren Inhalt beschreiben (es sei denn, sie sind rein dekorativ, dann alt="").
- Links: Alle Links müssen einen klaren Linktext haben, der ihren Zweck auch außerhalb des visuellen Kontexts verständlich macht (z.B. „Mehr erfahren über Produkt X“ statt nur „Mehr erfahren“).
- Formularfelder: Sollten Formularfelder in einem Slider liegen, müssen auch diese die Richtlinien für barrierefreie Formulare einhalten (sichtbare Labels, korrekte Verknüpfungen etc.).

Der Slider ist nur der Container; die Inhalte darin müssen die gleichen hohen Standards der Barrierefreiheit erfüllen, als stünden sie frei auf der Seite.