SOP 30

Mehrstufige Prozesse

Projektleitung | Geschäftsführung | IT | Recht

So machst du deine wichtigsten Prozesse für alle durchführbar

Weil Prozesse wie Buchung, Checkout, Leadgenerierung oder Registrierung die Kernfunktionen deiner Website sind – und genau hier entscheidet sich, ob Nutzer:innen aktiv teilnehmen können. Barrierefreiheit endet nicht beim Lesbaren, sie muss auch durchführbar sein.

Nur wenn Nutzer:innen barrierefrei durch deine Prozesse geführt werden, ist dein Angebot EAA-konform und nicht diskriminierend. Dies betrifft besonders Menschen mit motorischen oder kognitiven Einschränkungen sowie Screenreader-Nutzer:innen. Und nur dann kannst du die maximalen Conversions und deine Unternehmensziele überhaupt erreichen.

Wann musst du das machen?

- Bei der Erstellung oder dem Relaunch von Prozessen mit mehreren Schritten (z. B. Checkout, Event-Buchung, Bewerbung).
- Im Rahmen von WCAG/EAA-Audits.
- Wenn Beschwerden oder hohe Absprungraten auf bestimmten Schritten auftreten.

Welches Gesetz verlangt das?

- EAA (European Accessibility Act)
- WCAG 2.1 / 2.2 , insbesondere Kriterien wie:
- 1.3.1 Info and Relationships
- 2.4.6 Headings and Labels
- 3.3.1 Error Identification
- 3.3.2 Labels or Instructions
- 3.3.3 Error Suggestion
- 3.3.4 Error Prevention
- 4.1.3 Status Messages

Der Prozess im Detail

1 🟧 **Phase 1: Analysiere die Prozessstruktur**

Wie immer gilt: Bevor du ins Detail gehst, musst du den gesamten Prozess verstehen und in logische Einheiten zerlegen. Je komplexer der Prozess, desto klarer muss die Führung sein. So gehst du dabei vor:

- Zerlege den Prozess: Gliedere den Gesamtprozess in klare, einzelne Schritte (z. B. Warenkorb → Adresseingabe → Zahlungsmittel → Zusammenfassung → Bestätigung).
- Erstelle eine Fluss-Skizze: Visualisiere den Ablauf, zum Beispiel mit Tools wie Miro oder Lucidchart.
- Bedenke die Navigation: Achte darauf, ob Nutzer:innen Schritte überspringen oder zu vorherigen Schritten zurückspringen können.

2 🟨 Phase 2: Gestalte jeden einzelnen Schritt barrierefrei

Wende nun für jeden Schritt im Prozess konsequent die folgenden Gestaltungsprinzipien an:

- Verwende eindeutige Überschriften: Jeder Schritt braucht eine klare, semantisch korrekte Überschrift (z. B. <h2> für „Lieferadresse“).
- Gib den Nutzern Kontext: Zeige den Fortschritt an, sodass Nutzer:innen immer wissen, wo sie sich befinden (z. B. mit einer Fortschrittsanzeige: „Schritt 2 von 4“).
- Mache Labels immer sichtbar: Jedes Formularfeld muss ein permanent sichtbares Label haben, das per „label for“ korrekt verknüpft ist.
- Kennzeichne Pflichtfelder: Markiere Pflichtfelder sowohl visuell (z. B. mit „*“) als auch technisch (mit aria-required=“true“).
- Platziere Hilfetexte zugänglich: Platziere Hilfetexte direkt beim Feld und nicht nur als Tooltip, der bei Hover erscheint.
- Beschrifte deine Buttons klar: Verwende aussagekräftige Button-Texte wie „Weiter zur Zahlungsart“ anstelle eines generischen „Weiter“.

3 🟩 Phase 3: Stelle sicher, dass Fehlermeldungen und das Feedback auf die Aktion verständlich sind

Nutzer:innen müssen sofort und klar verstehen, was falsch gelaufen ist oder ob eine Aktion erfolgreich war.

- Erkläre Fehler im Text: Gib präzise Fehlermeldungen aus, z. B. „PLZ muss fünfstellig sein“.
- Verknüpfe die Fehlermeldung: Verknüpfe die Fehlermeldung technisch mit dem betroffenen Feld, z. B. über „aria-describedby“.
- Bestätige den Erfolg: Zeige nach dem Absenden eine klar verständliche Erfolgsmeldung an, die mit role=“status“ ausgezeichnet ist, damit Screenreader sie wahrnehmen.

4 🟥 Phase 4: Führe Tastatur- und Screenreader-Test durch

Wie immer musst du schließlich auch die theoretische Umsetzung in der Praxis überprüfen. Das machst du, indem du folgendermaßen vorgehst:

- Checke die Tab-Reihenfolge: Prüfe, ob die Navigation mit der Tab-Taste logisch ist und der Fokus nicht springt.
- Achte auf vollständige Bedienbarkeit: Alle Elemente (Buttons, Dropdowns, modale Fenster etc.) müssen mit der Tastatur (Tab, Enter, Space, Esc) erreichbar und steuerbar sein.
- Prüfe den sichtbaren Fokusindikator: Das jeweils aktive Element muss immer klar und deutlich erkennbar sein.
- Mache einen Screenreader-Test: Überprüfe die Leserichtung und die Verständlichkeit mit Tools wie NVDA oder VoiceOver.

5 🟦 Phase 5: Dokumentiere und etabliere eine kontinuierliche Pflege

Barrierefreiheit, Datenschutz und Datensicherheit gehen Hand in Hand. In der Praxis heißt das:

- Verlinke deine Datenschutzerklärung: Die DSGVO-konforme Datenschutzerklärung muss vor dem Login einfach erreichbar sein.
- Datensparsamkeit: Fordere bei der Registrierung nur die absolut notwendigen Felder (E-Mail & Passwort) als Pflichtfelder.
- Mache 2FA zugänglich: Wenn eine Zwei-Faktor-Authentisierung angeboten wird, muss diese barrierefrei sein (z. B. eine Alternative zur reinen App-Nutzung per E-Mail anbieten).
- Kommuniziere Timeouts: Informiere Nutzer:innen, wenn sie aufgrund von Inaktivität abgemeldet wurden.

Ergebnis

Am Ende dieser SOP hast du:

- Alle interaktiven Prozesse strukturiert und nachvollziehbar gestaltet.
- Das Fehlerhandling barrierefrei umgesetzt.
- Die Prozesse systematisch getestet und dokumentiert.
- Eine solide Grundlage für EAA-Nachweise bei Prüfungen geschaffen.

Um Login, Registrierung & Passwort-Reset blitzschnell barrierefrei zu gestalten, stell dir bei jedem Schritt die zentrale Frage:

„Kann das JEDE:R mit Tastatur und Screenreader problemlos nutzen?“

Wenn die Antwort NICHT SOFORT „Ja“ ist, dann liegt hier dein Optimierungspotenzial.

Umsetzungs-Template: Barrierefreie Prozesse & Formulare

Projekt/Prozessbezeichnung: [Hier den Namen des zu gestaltenden Prozesses eintragen, z.B. „Registrierungsprozess“, „Checkout-Flow“]

Verantwortliche(r) Projektmanager:in/Produktverantwortliche(r): [Dein Name] Verantwortliche(r) Entwicklung/ Umsetzung: [Name des/der Entwickler:in/Teams] Datum der Anfrage/Start: [Datum] Geplantes Go-Live: [Datum]

Phase 1: Prozessstruktur-Analyse (Ausfüllen durch PM/PO)

- Prozess in logische Schritte zerlegt?
- Ja
- Nein
- Falls Ja: Kurze Liste der Hauptschritte (z.B. Warenkorb > Adresse > Zahlung > Bestätigung):
- [Schritt 1]
- [Schritt 2]
- [Schritt 3]
- […]
- Fluss-Skizze erstellt und verlinkt?
- Ja, Link zur Skizze: [Link hier einfügen]
- Nein, wird nachgeholt bis: [Datum]
- Navigation (Vor/Zurück/Überspringen) bedacht?
- Ja, Navigation ist klar definiert.
- Nein, muss noch definiert werden für: [Schritte/Punkte]

Phase 2: Gestaltung jedes einzelnen Schrittes (Anweisung an Entwicklung)

Für JEDEN Schritt des oben definierten Prozesses sicherstellen, dass:

- Eindeutige Überschriften (

) verwendet werden:

- Checklist-Item für Entwickler: Überschriften für jeden Schritt prüfen.
- Fortschrittsanzeige implementiert ist (z.B. „Schritt X von Y“):
- Checklist-Item für Entwickler: Fortschrittsanzeige für den gesamten Flow implementieren.
- Labels immer sichtbar und korrekt mit „label for“ verknüpft sind:
- Checklist-Item für Entwickler: Jedes Formularfeld auf sichtbares, korrekt verknüpftes Label prüfen.
- Pflichtfelder visuell („*“) UND technisch (aria-required=“true“) markiert sind:
- Checklist-Item für Entwickler: Alle Pflichtfelder entsprechend kennzeichnen.
- Hilfetexte DIREKT beim Feld platziert sind (nicht nur als Tooltip):
- Checklist-Item für Entwickler: Hilfetexte auf Zugänglichkeit prüfen und ggf. anpassen.
- Button-Texte klar und aussagekräftig sind (z.B. „Weiter zur Zahlung“ statt „Weiter“):
- Checklist-Item für Entwickler: Alle Button-Texte im Prozess auf Klarheit prüfen und anpassen.

Phase 3: Fehlermeldungen & Feedback (Anweisung an Entwicklung)

- Fehlermeldungen präzise und im Text erklärt werden (z.B. „PLZ muss fünfstellig sein“):
- Checklist-Item für Entwickler: Alle möglichen Fehlermeldungen auf Präzision und Klarheit prüfen.
- Fehlermeldungen technisch mit betroffenem Feld verknüpft sind (aria-describedby):
- Checklist-Item für Entwickler: Technische Verknüpfung der Fehlermeldungen sicherstellen.
- Erfolgsmeldungen klar verständlich sind und mit role=“status“ ausgezeichnet werden:
- Checklist-Item für Entwickler: Erfolgsmeldungen implementieren und semantisch korrekt auszeichnen.

Phase 4: Tastatur- & Screenreader-Tests (Durchführung und Dokumentation durch Tester/QA)

- Tab-Reihenfolge geprüft und logisch?
- Ja
- Nein, Probleme in [Schritt/Bereich]
- Alle Elemente mit Tastatur bedienbar (Tab, Enter, Space, Esc)?
- Ja
- Nein, Probleme in [Schritt/Element]
- Sichtbarer Fokusindikator immer klar erkennbar?
- Ja
- Nein, Probleme in [Schritt/Element]
- Screenreader-Test durchgeführt (NVDA/VoiceOver)?
- Ja, keine Auffälligkeiten
- Ja, Probleme in [Schritt/Bereich], Details siehe [Link zum Testbericht]

Phase 5: Dokumentation & Kontinuierliche Pflege (Verantwortung: PM/PO & Entwicklung)

- Datenschutzerklärung vor dem Login/Formular einfach erreichbar und verlinkt?
- Ja, Link zur Datenschutzerklärung: [Link hier einfügen]
- Nein, muss noch integriert werden.
- Datensparsamkeit bei Pflichtfeldern umgesetzt (E-Mail, Passwort Minimum)?
- Ja
- Nein, folgende Felder sind noch Pflicht: [Liste unnötiger Pflichtfelder]
- 2FA (falls angeboten) barrierefrei umgesetzt (z.B. E-Mail-Alternative zur App)?
- Ja
- N/A (nicht angeboten)
- Nein, muss noch umgesetzt werden.
- Timeouts klar kommuniziert (wenn Nutzer:in abgemeldet)?
- Ja
- Nein, muss noch implementiert werden.

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

Fragen & Antworten

Warum ist die korrekte Überschriftenhierarchie so wichtig, und was passiert, wenn ich das ignoriere?

Die korrekte Überschriftenhierarchie (<h1> bis <h6>) ist fundamental für die Barrierefreiheit, besonders für Nutzende von Screenreadern . Stell dir vor, du scannst eine Webseite: Du suchst nach Überschriften, um dich schnell zu orientieren. Screenreader-Nutzende tun das auch, indem sie die Überschriftenliste einer Seite aufrufen. Wenn du die Hierarchie nicht korrekt einhältst (z.B. <h2> nach <h4> oder gar keine Überschriften verwendest), können Screenreader-Nutzende die Seitenstruktur nicht erfassen . Sie wissen nicht, wo ein neuer Abschnitt beginnt oder wie die Inhalte zueinander in Beziehung stehen. Das führt zu Verwirrung und erschwert die Navigation massiv. Es ist vergleichbar mit einem Buch ohne Inhaltsverzeichnis oder Kapitelüberschriften – extrem mühsam, sich zurechtzufinden.

Wie kann ich sicherstellen, dass mein %22sichtbarer Fokusindikator%22 immer klar und deutlich erkennbar ist, auch bei komplexen Designs?

Der sichtbare Fokusindikator ist entscheidend für Nutzende, die mit der Tastatur navigieren. Oft ist der Standard-Browser-Fokusring (ein blauer oder gestrichelter Rahmen) bei komplexen oder farbigen Designs kaum sichtbar. Um das zu verbessern, du, solltest du im CSS gezielt den :focus -Zustand deiner Elemente gestalten. Hier ein paar Tipps:

- Deutlicher Kontrast: Wähle Farben für den Fokusindikator, die einen hohen Kontrast zum Hintergrund und zum Element selbst aufweisen.
- Formänderung: Füge zum Rahmen noch eine leichte Vergrößerung, eine andere Hintergrundfarbe oder eine dickere Umrandung hinzu.
- Kein outline: none; : Vermeide es unbedingt, outline: none; in deinem CSS zu verwenden, da dies den Fokusindikator komplett entfernt und Elemente für Tastaturnutzende unsichtbar macht. Eine gute Praxis ist, den Fokusindikator immer so zu gestalten, dass er sowohl auf dunklen als auch auf hellen Hintergründen deutlich sichtbar ist.

Die SOP erwähnt %22role='status'%22 für Erfolgsmeldungen. Gibt es ähnliche %22ARIA-Rollen%22, die ich für andere dynamische Inhalte oder Statusänderungen nutzen sollte?

Ja, absolut! ARIA-Rollen (Accessible Rich Internet Applications) sind essenziell, um Screenreadern und anderen assistiven Technologien Informationen über dynamische Inhalte zu geben, die sich auf der Seite ändern, ohne dass die Seite neu geladen wird. Neben role="status" für nicht-unterbrechende Statusmeldungen gibt es weitere nützliche Rollen:

- role="alert" : Für dringende, zeitkritische und potenziell störende Nachrichten (z.B. „Ihre Sitzung läuft in 30 Sekunden ab“).
- role="alertdialog" : Für modale Dialoge, die eine sofortige Nutzeraktion erfordern (z.B. eine Bestätigung vor dem Löschen).
- role="progressbar" : Für Ladebalken oder Fortschrittsanzeigen, um Screenreadern den aktuellen Status zu vermitteln (oft in Kombination mit aria-valuenow , aria-valuemin , aria-valuemax).
- role="live" regions (aria-live="polite" oder aria-live="assertive"): Diese sind mächtig und können ganze Bereiche der Seite als „Live-Region“ markieren, sodass Screenreader automatisch Änderungen in diesem Bereich vorlesen. polite liest Änderungen vor, wenn der Screenreader „Zeit hat“ (nicht störend), assertive unterbricht sofort.

Der korrekte Einsatz dieser Rollen stellt sicher, dass Nutzende über wichtige dynamische Änderungen informiert werden, auch wenn sie diese nicht visuell wahrnehmen können.

•

Meine Formulare sind sehr lang. Gibt es eine Empfehlung, wie ich die Formularfelder gruppieren oder organisieren sollte, um die Barrierefreiheit zu verbessern?

Ja, für lange Formulare ist Struktur extrem wichtig. Du kannst, du, folgende Techniken nutzen:

- fieldset und legend : Dies sind HTML-Elemente, die logisch zusammengehörige Formularfelder gruppieren (fieldset) und dieser Gruppe eine Überschrift (legend) geben. Zum Beispiel: <fieldset><legend>Ihre Kontaktdaten</legend> ... Formularfelder ... </fieldset> . Screenreader lesen die Legende vor, wenn sie in den Feldset-Bereich kommen, und geben so Kontext.
- Visuelle Gruppierung: Auch visuell sollten zusammengehörige Felder durch Abstände oder Boxen gruppiert werden.
- Schrittweise Offenlegung (Progressive Disclosure): Bei sehr komplexen Formularen kannst du überlegen, Felder oder ganze Abschnitte erst dann anzuzeigen, wenn sie relevant werden (z.B. Zusatzfelder für „Andere Versandadresse“ erst bei Klick auf eine Checkbox). Dies reduziert die kognitive Last.

Eine gute Strukturierung hilft allen Nutzenden, das Formular besser zu überblicken und auszufüllen.

•

Stichwort %22Datensparsamkeit%22. Wie weit kann ich damit gehen, und gibt es Fälle, in denen mehr Daten gesammelt werden müssen, ohne die Barrierefreiheit zu beeinträchtigen?

Datensparsamkeit ist ein Grundprinzip der DSGVO und gleichzeitig ein Plus für die Nutzerfreundlichkeit. Du solltest nur die Daten als Pflichtfelder abfragen, die unbedingt notwendig sind, um den primären Zweck des Formulars (z.B. Registrierung, Bestellung) zu erfüllen.

- Beispiel Registrierung: Oft reichen E-Mail und Passwort als Pflichtfelder. Name, Adresse, Telefonnummer könnten optional sein oder erst später im Checkout-Prozess abgefragt werden.
- Beispiel Bestellung: Hier sind natürlich Adressdaten und Zahlungsinformationen Pflicht, da sie für die Abwicklung des Kaufvertrags notwendig sind.

Wichtig ist, dass diese notwendigen Felder immer klar als Pflichtfelder gekennzeichnet und zugänglich sind, wie in deiner SOP beschrieben. Wenn du mehr Daten benötigst, die nicht zwingend für den primären Zweck sind, mache sie immer optional und erkläre gegebenenfalls, wofür sie verwendet werden (z.B. „Telefonnummer für Rückfragen zur Lieferung (optional)“). Das beeinträchtigt die Barrierefreiheit nicht, sondern erhöht die Nutzerakzeptanz und die Datenschutzkonformität.