Richtlinien für barrierefreie Webinhalte

Diese Checkliste bietet eine praxisorientierte Übersicht der **Web Content Accessibility Guidelines (WCAG) 2.2**. Sie fasst die wichtigsten Erfolgskriterien zusammen, ergänzt durch kurze Erklärungen und Beispiele zur barrierefreien Gestaltung von Webinhalten.

Die Kriterien sind entsprechend ihrer Konformitätsstufen A und AA gekennzeichnet. Für Websites/Online-Shops wird in der Regel die Einhaltung der Stufe AA empfohlen bzw. gesetzlich gefordert.

Hinweis: Ab Juni 2025 ist gemäß EU-Vorgaben die WCAG 2.1 verpflichtend. Es wird jedoch erwartet, dass Ende 2025/Anfang 2026 die WCAG 2.2 als verbindlicher Standard eingeführt wird. Daher orientiert sich diese Übersicht bereits an der Version 2.2.

Zur originalen Verfassung: https://www.w3.org/TR/WCAG22/

1. Wahrnehmbar

Informationen und Benutzeroberflächenkomponenten müssen den Benutzern auf eine für sie verständliche Weise präsentiert werden.

📌 **1.1.1 Nicht-Text-Inhalt [A]** *Alle wichtigen Bilder, Icons und Grafiken müssen als Textalternativen verfügbar sein.* 💡 Beispiel: Produktbild mit alt="Rote Sportjacke Vorderansicht" 💡 Beispiel: Icon für „Warenkorb“ mit aria-label="Warenkorb öffnen"	
📌 **1.2.1 Nur Audio und nur Video (aufgezeichnet) [A]** *Für reine Audio- oder reine Videoaufzeichnungen muss eine inhaltlich gleichwertige Text-Alternative vorhanden sein.* 💡 Beispiel: Textversion eines Podcast-Inhalts 💡 Beispiel: Beschreibung eines erklärenden Animationsvideos	
📌 **1.2.2 Untertitel (aufgezeichnet) [A]** *Vorab aufgezeichnete Videos mit Ton müssen Untertitel enthalten, damit auch Menschen ohne Hörvermögen den Inhalt verstehen.* 💡 Beispiel: Erklärvideo mit eingeblendeten Untertiteln 💡 Beispiel: YouTube-Video mit manuell gepflegten Untertiteln	
📌 **1.2.3 Audiodeskription oder Medienalternative (aufgezeichnet) [A]** *Für Videos mit wichtigen visuellen Inhalten muss es eine Audiodeskription oder ein vollständiges Transkript geben.* 💡 Beispiel: Video mit einer Tonspur, die Handlungen beschreibt 💡 Beispiel: Vollständiges Text-Transkript auf der Website	

📌 **1.2.4 Untertitel (Live) [AA]** *Live-Videos mit Ton müssen mit Live-Untertiteln versehen werden.* 💡 Beispiel: Livestream mit eingeblendeten Echtzeit-Untertiteln 💡 Beispiel: Webinar mit automatischer Untertitelung	
📌 **1.2.5 Audiodeskription (aufgezeichnet) [AA]** *Für vorab aufgezeichnete Videos müssen zusätzliche Audiodeskriptionen verfügbar sein, wenn visuelle Inhalte allein nicht ausreichen.* 💡 Beispiel: Film mit separater Tonspur für visuelle Beschreibungen 💡 Beispiel: Schulungsvideo mit Sprecher, der auch visuelle Abläufe erklärt	
📌 **1.3.1 Info und Beziehungen [A]** *Dieses Kriterium stellt sicher, dass Inhalte nicht nur visuell verständlich sind, sondern auch im Code richtig abgebildet werden. Struktur und Beziehungen – etwa zwischen einer Beschriftung und einem Eingabefeld – sollen für Screenreader und andere Hilfstechnologien programmgesteuert erkennbar sein.* 💡 Beispiel: <label for="email">E-Mail-Adresse</label> 💡 Beispiel: <fieldset> und <legend> bei Auswahlgruppen wie Lieferoptionen	
📌 **1.3.2 Sinnvolle Reihenfolge [A]** *Wenn die Reihenfolge der Inhalte wichtig für das Verständnis ist, muss diese im Code nachvollziehbar und korrekt abgebildet sein, damit Nutzern von Screenreadern die Inhalte logisch vorgelesen werden.* 💡 Beispiel: Produktname → Preis → „In den Warenkorb“-Button 💡 Beispiel: Menüpunkt 1 → Menüpunkt 2 → Suchfeld	
📌 **1.3.3 Sensorische Eigenschaften [A]** *Hinweise dürfen nicht nur durch Farbe, Form oder Position gegeben werden.* 💡 Beispiel: „Pflichtfelder sind mit * gekennzeichnet“ statt „rote Felder ausfüllen“ 💡 Beispiel: „Wählen Sie die Option mit dem Häkchen“ statt „oben rechts“	
📌 **1.3.4 Orientierung [AA]** *Inhalte dürfen nicht auf eine bestimmte Anzeigeausrichtung (z. B. nur Hochformat) beschränkt sein – außer, diese ist unbedingt erforderlich (z. B. VR-Anwendungen)* 💡 Beispiel: Eine Website funktioniert sowohl im Hoch- als auch im Querformat 💡 Beispiel: Inhalte passen sich automatisch an, statt Hinweis „Bitte drehen Sie Ihr Gerät“	

📌 **1.3.5 Eingabezweck identifizieren [AA]** *Der Zweck von Eingabefeldern (z. B. Name, E-Mail) muss programmgesteuert erkennbar sein, z. B. über HTML-Attribute wie autocomplete.* 💡 Beispiel: <input type="text" autocomplete="email"> für ein E-Mail-Feld 💡 Beispiel: <input type="text" autocomplete="given-name"> für den Vornamen	
📌 **1.4.1 Verwendung von Farbe [A]** *Informationen dürfen nicht ausschließlich über Farbe vermittelt werden.* 💡 Beispiel: Fehlermeldung in Rot + Text: „Bitte geben Sie ein gültiges Passwort ein“ statt nur roter Markierung 💡 Beispiel: Lagerstatus mit Text: „Verfügbar“ statt nur grünem Punkt	
📌 **1.4.2 Audiosteuerung [A]** *Wenn Audio automatisch abgespielt wird und länger als 3 Sekunden dauert, muss es eine Möglichkeit geben, den Ton unabhängig von der Systemlautstärke zu stoppen, pausieren oder leiser zu stellen.* 💡 Beispiel: Eine Webseite spielt automatisch Hintergrundmusik ab, bietet aber eine deutliche Möglichkeit, die Musik zu stoppen (z. B. sichtbares „🔇 Audio aus“-Symbol) 💡 Beispiel: Die Seite spielt nur dann einen Sound ab, wenn der Nutzer aktiv auf „Play“ klickt	
📌 **1.4.3 Kontrast (Minimum) [AA]** *Text muss ausreichend Kontrast zum Hintergrund haben (mind. 4.5:1).* *Großformatiger Text (min. 18 Punkt bzw. 14 Punkt in Fett) benötigen einen Kontrast von 3:1.* *Für Bilder, die ausschließlich dekorativ sind, für niemanden sichtbar sind oder Teil eines größeren Bildes mit bedeutenden Inhalten sind, gelten keine spezifischen Kontrastanforderungen.* 💡 Beispiel: Kontrast checken mit Tools wie der Color Contrast Checker von Coolors. 💡 Beispiel: Ein Telefon-Icon, das neben einer Telefonnummer angezeigt wird, muss nur dann einen ausreichenden Kontrast aufweisen, wenn es interaktiv ist, also eine Handlungsaufforderung wie das Klicken auf die Telefonnummer darstellt. Wenn das Icon jedoch nur dekorativ ist und keine zusätzliche Information vermittelt, gelten keine speziellen Kontrastanforderungen für das Icon selbst. In diesem Fall liegt der Fokus auf dem ausreichenden Kontrast der Telefonnummer.	
📌 **1.4.4 Textgröße änderbar [AA]** *Inhalte müssen bei 200 % Zoom noch vollständig nutzbar sein.* 💡 Beispiel: Responsives Design sorgt dafür, dass Text auch bei 200 % Zoom lesbar bleibt. 💡 Beispiel: Flexible Layouts mit CSS (z. B. em, rem statt px).	

📌 **1.4.5 Bilder von Text [AA]** *Vermeide es, Text als Bild darzustellen – nutze stattdessen richtigen Text, der sich anpassen lässt (z. B. vergrößern, Farben ändern). Ausnahmen erlaubt, wenn die Darstellung entscheidend ist, z. B. bei Logos, Kunstwerken oder typografischen Beispielen.* 💡 Beispiel: Text auf einem Bild sollte man vermeiden – stattdessen den Text direkt als HTML-Text einfügen, damit er anpassbar bleibt. 💡 Beispiel: Logo mit Text als Bild – ist erlaubt, da Darstellung wesentlich ist.	
📌 **1.4.10 Reflow [AA]** *Texte und Inhalte müssen auch bei 400 % Zoom oder auf kleinen Bildschirmen nutzbar bleiben – ohne horizontales Scrollen.* *Zweidimensionales Scrollen ist erlaubt, wenn Inhalte aus funktionalen oder inhaltlichen Gründen ein zweidimensionales Layout erfordern, z. B. Karten, Diagramme oder Tabellen.* 💡 Beispiel: Die Produktdetailseite passt sich bei starker Vergrößerung oder auf Smartphones automatisch an. 💡 Beispiel: Kein horizontaler Scrollbalken nötig, weil Texte und Elemente umfließen.	
📌 **1.4.11 Nicht-Text-Kontrast [AA]** *UI-Elemente (z. B. Formfelder, Buttons) und wichtige grafische Informationen müssen sich deutlich vom Hintergrund abheben – mit einem Kontrastverhältnis von mindestens 3:1. Betroffen sind also alle grafischen Objekte, die visuell erkannt werden müssen, um eine Funktion zu verstehen (z. B. Lupe-Icon für Suchfunktion).* *Ausnahmen sind z. B. inaktive Elemente (z. B. inaktive Formularfelder), reine Deko-Grafiken ode vom Browser definierte Standard-Darstellungen.* 💡 Beispiel: Ein Symbol-Icon (z. B. Lupe für Suche) hebt sich klar vom Hintergrund ab. 💡 Beispiel: Deutlich sichtbarer Button mit Hover-Effekt.	
📌 **1.4.12 Textabstand [AA]** *Textabstände müssen anpassbar sein, um die Lesbarkeit zu verbessern. Der Entwickler muss die Abstände mit relativen Einheiten wie rem oder em festlegen, damit Benutzer die Möglichkeit haben, Zeilenhöhe (mindestens 1,5-fache Schriftgröße), Absatzabstand (mindestens doppelt so groß wie Schriftgröße), Buchstabenabstand (mindestens 0,12-mal Schriftgröße) und Wortabstand (mindestens 0,16-mal Schriftgröße) anzupassen, ohne dass der Inhalt unleserlich wird. Eine direkte Anpassung durch den Entwickler ist nicht erforderlich.* 💡 Beispiel: Nutzer können den Zeilenabstand auf 1,5-fache Schriftgröße erhöhen, ohne dass der Text schwer lesbar wird. 💡 Beispiel: Wenn der Abstand zwischen Absätzen verdoppelt wird, bleibt das Layout klar und lesbar.	

📌 **1.4.13 Inhalt bei Hover oder Fokus [AA]** *Zusätzliche Inhalte, die bei Hover oder Fokus erscheinen (z. B. Tooltips), müssen steuerbar bleiben. Der Benutzer muss den Inhalt bei Bedarf schließen können, ohne den Mauszeiger zu bewegen oder den Fokus zu ändern, außer wenn der Inhalt einen Fehler anzeigt oder andere Inhalte verdeckt. Der Inhalt sollte sichtbar bleiben, bis der Hover- oder Fokus-Trigger entfernt wird.* 💡 Beispiel: Ein Tooltip, der beim Hover eines Buttons erscheint, bleibt sichtbar, bis der Mauszeiger den Bereich verlässt. 💡 Beispiel: Ein Hinweistext bei der Eingabe von Formularfeldern erscheint beim Fokus auf das Feld und bleibt sichtbar, bis der Benutzer das Feld verlässt.	
2. Bedienbar **Benutzeroberflächenkomponenten und Navigation müssen bedienbar sein.**	
📌 **2.1.1 Tastaturbedienung [A]** *Alle Funktionen müssen über die Tastatur zugänglich und bedienbar sein, ohne dass spezielle Zeitvorgaben erforderlich sind (außer bei Eingabefunktionen, die vom Bewegungsweg wie z. B. Zeichenprogramme abhängen).* 💡 Beispiel: Mit „Tab“ kann der Nutzer durch Navigation, Produktauswahl, Warenkorb etc. springen. 💡 Beispiel: Mit der Enter-Taste wird ein Button aktiviert. 💡 Beispiel: In einem Online-Shop kann ein Produkt in den Warenkorb gezogen werden, was eine Mausbewegung erfordert. Diese Funktion kann nicht ohne weiteres mit der Tastatur durchgeführt werden, weil die Bewegung des Objekts von A nach B entscheidend ist. Daher stellt dies eine Ausnahme da.	
📌 **2.1.2 Keine Tastaturfalle [A]** *Der Tastaturfokus sollte nicht „festhängen“. Nutzer müssen in der Lage sein, mit der Tastatur problemlos durch alle Elemente zu navigieren.* 💡 Beispiel: Ein Popup-Fenster kann per „Escape“ oder durch Drücken der Tab-Taste verlassen werden. 💡 Beispiel: Der Filterbereich in einem Shop ist vollständig durch die Tab-Taste erreichbar, sodass der Nutzer ihn ohne Maus vollständig bedienen kann.	

📌 **2.1.4 Tastenkombinationen für Zeichen [A]** *Wenn eine Website Tastenkombinationen verwendet, die nur aus normalen Zeichen bestehen (wie Buchstaben, Zahlen oder Satzzeichen), kann es leicht passieren, dass man sie aus Versehen aktiviert – besonders bei Spracheingabe oder versehentlichen Tastenanschlägen. Damit das nicht passiert, muss es eine Möglichkeit geben, solche Tastenkombinationen zu deaktivieren, anzupassen oder sie nur bei aktivem Fokus auszulösen.* 💡 Beispiel: Manche Webseiten oder Programme nutzen Buchstaben-Tasten als Abkürzung – also, wenn man z. B. einfach „S“ drückt, wird etwas gespeichert oder eine Suche gestartet. Da das stören könnte, muss man diese Kombination deaktivieren oder ändern können. 💡 Beispiel: Ein Nutzer, der mit einer Tastatur arbeitet, drückt versehentlich die „S“-Taste und verliert dadurch den Fokus auf dem Text. Die Webseite ermöglicht es dem Nutzer, die Tastenkombination so zu ändern, dass sie eine zusätzliche „Strg“-Taste benötigt, um die ungewollte Aktivierung zu verhindern.	
📌 **2.2.1 Timing einstellbar [A]** *Wenn Inhalte zeitlich begrenzt sind (z. B. bei einer Sitzung oder Animation), muss Nutzer:innen genug Zeit zur Verfügung stehen, um die Aufgabe ohne unnötigen Druck abzuschließen. Sie sollen das Zeitlimit deaktivieren, verlängern oder anpassen können – außer das Zeitlimit ist aus bestimmten Gründen unvermeidlich (z. B. bei Auktionen oder aus Sicherheitsgründen).* 💡 Beispiel: Ein Online-Formular läuft nach 10 Minuten Inaktivität automatisch ab. Vor dem Ablauf erscheint ein Hinweis mit der Option, die Sitzung per Klick, um weitere 10 Minuten zu verlängern – dies ist bis zu zehnmal möglich. 💡 Beispiel: Eine Webseite zeigt automatisch rotierende Schlagzeilen. Der Nutzer kann über eine Schaltfläche die Geschwindigkeit anpassen oder die Rotation komplett pausieren, um in Ruhe lesen zu können.	
📌 **2.2.2 Pause, Stopp, Ausblenden [A]** *Bewegte, blinkende, scrollende oder automatisch aktualisierte Inhalte, die automatisch starten, länger als 5 Sekunden dauern und gleichzeitig mit anderen Inhalten angezeigt werden, müssen durch den Nutzer anhaltbar, stoppbar oder ausblendbar sein – außer sie sind für die Funktion der Seite wesentlich.* 💡 Beispiel: Ein automatisch startender Bildslider auf der Startseite zeigt alle 4 Sekunden ein neues Bild. Der Slider enthält eine Pause-Schaltfläche, mit der Nutzer*innen den automatischen Ablauf anhalten können. 💡 Beispiel: Ein Newsticker läuft am oberen Seitenrand. Rechts daneben befindet sich ein „Ticker anhalten“-Button, mit dem die Bewegung gestoppt werden kann.	

📌 **2.3.1 Schwellenwert für drei Blitze oder weniger [A]** *Webseiten dürfen keine Inhalte enthalten, die mehr als dreimal pro Sekunde blinken oder blitzen, es sei denn, das Blinken liegt unterhalb der Grenzwerte, die als sicher gelten. Ziel ist es, lichtempfindliche Anfälle (z. B. epileptische Anfälle) zu vermeiden.* 💡 Beispiel: Ein Online-Spiel enthält eine Explosion, die aufleuchtet. Die Animation wurde so angepasst, dass nicht mehr als drei Blitze pro Sekunde sichtbar sind – dadurch bleibt sie unter dem kritischen Schwellenwert. 💡 Beispiel: Eine Musik-Website zeigt eine Visualisierung mit Lichtblitzen. Der Entwickler reduziert die Intensität und Fläche der Blitze und stellt sicher, dass sie die 87.296 CSS-Pixel (10° Sichtfeld) nicht überschreiten – die Animation bleibt damit sicher für alle Nutzer*innen.	
📌 **2.4.1 Blöcke umgehen [A]** *Nutzerinnen müssen die Möglichkeit haben, wiederkehrende Inhalte wie Navigationsleisten oder Menüblöcke zu überspringen, um direkt zum Hauptinhalt einer Seite zu gelangen (sog. „Skip to Content“-Link)* 💡 Beispiel: Ein versteckter Link „Zum Hauptinhalt springen“, der direkt oben auf der Seite ist, aber erst sichtbar wird, wenn der Benutzer mit der Tab-Taste dorthin navigiert. 💡 Beispiel: Der Link „Zum Hauptinhalt springen“ wird immer angezeigt, sobald der Benutzer mit der Tab-Taste die Seite betritt.	
📌 **2.4.2 Seitentitel [A]** *Jede Seite braucht einen aussagekräftigen und beschreibenden Titel, der Thema oder Zweck der Seite klar angibt. Dieser Titel hilft allen Nutzer*innen bei der Orientierung, besonders wenn mehrere Tabs oder Seiten geöffnet sind.* 💡 Beispiel: Der Titel einer Seite über Heizungssysteme lautet im Browser-Tab: „Moderne Heizsysteme – Firma XY“ → Der Nutzer erkennt sofort, worum es auf der Seite geht. 💡 Beispiel: Eine Banking-Webanwendung zeigt im Titel: „Kontoauszug Dezember 2024 – Bank XYZ“ → So wissen Nutzer sofort, welche Funktion oder Ansicht sie gerade nutzen.	
📌 **2.4.3 Fokus-Reihenfolge [A]** *Interaktive Elemente erhalten den Fokus in einer logischen Reihenfolge, die Inhalt und Funktion unterstützt. Wenn Nutzer eine Seite mit der Tastatur durchlaufen (z. B. mit der Tab-Taste), sollten alle bedienbaren Elemente in einer sinnvollen Reihenfolge fokussiert werden – also so, wie sie inhaltlich und funktional zusammengehören.* 💡 Beispiel: Der Fokus sollte in einem Formular von oben nach unten durch die Eingabefelder geführt werden (z. B. Name → Adresse → Stadt → Postleitzahl → Absenden). 💡 Beispiel: In einer Tabelle sollte der Fokus zeilenweise oder spaltenweise von Zelle zu Zelle wandern, entsprechend der natürlichen Struktur der Daten.	

📌 **2.4.4 Linkzweck (im Kontext) [A]** *Links müssen verständlich beschriftet sein, sodass der Zweck jedes Links allein aus dem Linktext oder aus dem Linktext zusammen mit seinem durch Software bestimmten Kontext ermittelt werden kann.* 💡 Beispiel: „Mehr über Zahlungsarten erfahren“ statt „Hier klicken“. 💡 Beispiel: Der Text eines Links, der zu einer PDF-Datei führt, sollte den Zweck der Datei beschreiben, z. B. „Preisliste herunterladen“ anstelle von „Download“.	
📌 **2.4.5 Mehrere Möglichkeiten [AA]** *Es sollte auf einer Website mehrere Wege geben, um denselben Inhalt zu finden. Dadurch können Nutzer je nach ihren Bedürfnissen und Vorlieben auf verschiedene Arten durch die Website navigieren.* 💡 Beispiel: Auf einer Rezept-Website kannst du entweder nach „Suppe“ suchen oder „Suppe“ aus einer Liste von Kategorien auswählen, um zu den Rezepten zu gelangen. 💡 Beispiel: Eine Übersicht (z. B. Sitemap) mit Links zu allen Seiten der Website, um direkt zu dem gewünschten Inhalt zu springen.	
📌 **2.4.6 Überschriften und Beschriftungen [AA]** *Überschriften und Beschriftungen sollten klar und beschreibend sein, um den Nutzern zu helfen, das Thema oder den Zweck eines Inhalts zu verstehen.* 💡 Beispiel: Ein Leitfaden zum guten Schreiben hat klare und prägnante Abschnittsüberschriften wie z. B. „So schreiben Sie gut“, die den Inhalt der Abschnitte deutlich machen. 💡 Beispiel: Ein Formular fragt nach dem Vor- und Nachnamen des Nutzers. Die Felder sind klar mit „Vorname“ und „Nachname“ beschriftet.	
📌 **2.4.7 Sichtbarer Fokus [AA]** *Der aktuell fokussierte Bereich muss visuell hervorgehoben sein (Kontrast min. 3:1).* 💡 Beispiel: Schattierung oder stärkerer Rahmen um aktive Felder/Buttons. 💡 Beispiel: Wenn ein Benutzer in ein Formularfeld klickt oder mit der Tabulatortaste zum Feld navigiert, wird der Rahmen des Feldes dicker und in einer kontrastreicheren Farbe angezeigt, um den aktuellen Fokus visuell zu kennzeichnen.	
📌 **2.4.11 Fokus nicht verdeckt (Minimum) [AA]** *Der Fokus muss für Tastaturnutzer sichtbar bleiben und darf nicht durch andere Elemente verdeckt werden.* 💡 Beispiel: Der Fokus bleibt sichtbar, auch wenn ein Sticky Footer angezeigt wird. 💡 Beispiel: Wenn ein Benutzer in ein Textfeld klickt und dann ein Menü erscheint, wird das Textfeld nicht von dem Menü verdeckt, sodass der Benutzer immer noch sehen kann, was er gerade bearbeitet.	

📌 **2.5.1 Zeigergesten [A]** *Alle Funktionen, die mehrpunkt- oder pfadbasierte Gesten erfordern, sollten auch mit einem einzelnen Zeiger (z.B. einem Finger oder der Maus) bedient werden können, es sei denn, die Mehrpunkt- oder pfadbasierte Geste ist unbedingt erforderlich.* 💡 Beispiel: Ein Karten-Viewer auf einer Website lässt den Benutzer sowohl mit der Pinch-Geste (zwei Finger zum Vergrößern) als auch durch Klicken auf Schaltflächen zum Vergrößern und Verkleinern der Karte navigieren. 💡 Beispiel: Ein horizontaler Inhalts-Slider bietet die Möglichkeit, durch Wischen zu navigieren, erlaubt jedoch auch die Navigation durch Klicken auf Pfeilschaltflächen, die denselben Effekt haben.	
📌 **2.5.2 Zeigerstornierung [A]** *Dieses Erfolgskriterium zielt darauf ab, versehentliche oder ungewollte Aktionen mit Maus oder Touchscreen zu verhindern. Es verlangt, dass Interaktionen mit einem Zeiger (z. B. Maus oder Touch) entweder beim Loslassen (Up-Ereignis) abgeschlossen werden oder die Möglichkeit besteht, die Aktion rückgängig zu machen.* 💡 Beispiel: Bei einem einfachen Klick erfolgt die Aktion erst, wenn der Benutzer die Maustaste oder den Finger vom Bildschirm hebt. Dies gibt dem Benutzer die Möglichkeit, die Aktion abzubrechen, bevor sie abgeschlossen wird. 💡 Beispiel: Bei einer Drag-and-Drop-Interaktion wird das Element erst dann verschoben, wenn der Benutzer die Maustaste oder den Finger loslässt.	
📌 **2.5.3 Beschriftung im Namen [A]** *Beschriftung von Elementen muss mit ihrer technischen Bezeichnung übereinstimmen.* 💡 Beispiel: Ein Button mit dem sichtbaren Text „Warenkorb“ verwendet kein zusätzliches aria-label, da der sichtbare Text bereits als zugänglicher Name dient. 💡 Beispiel: Ein Warenkorb-Icon-Button ohne sichtbaren Text hat aria-label="Warenkorb" – der Labeltext entspricht damit der visuellen Bedeutung.	
📌 **2.5.4 Bewegungsbestätigung [A]** *Dieses Erfolgskriterium stellt sicher, dass Funktionen, die durch die Bewegung eines Geräts oder durch Benutzerbewegungen (wie Gesten) ausgelöst werden, auch über herkömmliche Benutzeroberflächenkomponenten gesteuert werden können.* 💡 Beispiel: Rückgängig durch Schütteln: Ein Benutzer kann durch Schütteln des Geräts eine Aktion rückgängig machen. Es gibt jedoch auch eine Schaltfläche „Rückgängig“, die diese Funktion auf herkömmliche Weise ausführt. 💡 Beispiel: Ein Benutzer kann das Gerät neigen, um zwischen Seiten zu wechseln. Alternativ gibt es auch Schaltflächen, um diese Navigation auszuführen.	

📌 **2.5.7 Schleppbewegungen [AA]** *Dieses Erfolgskriterium stellt sicher, dass alle Funktionen, die durch Ziehbewegungen (Drag-and-Drop) bedient werden, auch ohne Ziehen mit einem einzelnen Zeiger (z.B. durch Tippen oder Klicken) erreichbar sind, es sei denn, das Ziehen ist zwingend erforderlich oder wird durch den Benutzeragenten (z.B. Browser) bestimmt.* 💡 Beispiel: Ein Benutzer kann ein Element in einer Liste durch Ziehen an eine andere Position verschieben. Als Alternative gibt es Schaltflächen, um das Element durch Klicken nach oben oder unten zu verschieben. 💡 Beispiel: Ein Farbrad ermöglicht es, eine Farbe durch Ziehen eines Markers auszuwählen. Alternativ können Benutzer einen Farbwert durch Klicken auf ein beliebiges Feld im Farbrad auswählen.	
📌 **2.5.8 Zielgröße (Minimum) [AA]** *Interaktive Elemente (z. B. Buttons) müssen mindestens 24x24px groß sein. Wenn Ziele kleiner sind, muss entweder ausreichend Abstand zu anderen Zielen bestehen – konkret: ein gedachter Kreis mit 24px Durchmesser darf sich nicht mit dem Kreis eines anderen Ziels überschneiden – oder es muss eine alternative Möglichkeit zur Interaktion angeboten werden.* 💡 Beispiel: Ein kleines „x“-Symbol zum Schließen eines Popups ist nur 16x16px groß, hat aber rundherum 12px Abstand zu anderen Bedienelementen. Dadurch überschneiden sich die imaginären 24px-Kreise nicht – das Kriterium ist erfüllt. 💡 Beispiel: Ein Formular enthält mehrere Schaltflächen, die kleiner als 24 x 24 CSS-Pixel sind. Wenn diese Schaltflächen jedoch genügend Abstand voneinander haben (mindestens 24 CSS-Pixel), erfüllt es das Kriterium.	
3. Verständlichkeit **Informationen und Bedienung der Benutzeroberfläche müssen verständlich sein.**	
📌 **3.1.1 Sprache der Seite [A]** *Die Hauptsprache der Seite muss im Code korrekt angegeben sein, damit Technologien wie Screenreader die Inhalte richtig vorlesen oder darstellen können. Die Sprache wird durch das lang-Attribut am <html>-Tag definiert.* 💡 *Beispiel:* <html lang="de"> für deutschsprachige Shops	
📌 **3.1.2 Sprache der Teile [AA]** *Wenn innerhalb einer Webseite Textabschnitte oder Wörter in einer anderen Sprache als der Hauptsprache erscheinen, muss dies im Code deutlich ausgezeichnet werden. Nur so können Screenreader und andere Technologien korrekt reagieren – z. B. durch richtige Aussprache.* *Wenn sich die Sprache also innerhalb des Inhalts ändert, muss dies ausgezeichnet werden – außer bei Eigennamen, Fachbegriffen, eingedeutschten Wörtern oder solchen, die zur Umgangssprache des umgebenden Textes gehören.* 💡 Beispiel: Auf einer deutschen Webseite steht: Sie sagte „C’est la vie“. Im Code muss der französische Teil wie folgt gekennzeichnet werden: C’est la vie. 💡 Beispiel: Ebenso müssen die Sprachen innerhalb einer Sprachauswahl mit dem richtigen lang versehen werden, z. B. English: lang=“en“ oder lang=“fr“.	

📌 **3.2.1 Im Fokus [A]** *Wenn Nutzer mit der Tastatur oder Maus durch eine Webseite navigieren, darf sich beim Fokussieren eines Elements (z. B. eines Eingabefelds oder Buttons) nicht plötzlich etwas verändern – zum Beispiel eine neue Seite öffnen oder ein Fenster erscheinen. Nur durch das aktive Auslösen (z. B. per Klick oder Enter) darf sich der Kontext verändern.* 💡 Beispiel: Ein Benutzer springt mit der Tabulatortaste auf ein Dropdown-Menü mit verschiedenen Seiten. Erst wenn er eine Auswahl trifft und Enter drückt, wird die neue Seite geladen. 💡 Beispiel: Ein Nutzer springt mit der Tab-Taste in ein Eingabefeld, und sofort öffnet sich ein Hilfe-Popup mit Tipps. Das ist nicht zulässig, da sich der Kontext allein durch das Erreichen des Fokus verändert hat – ohne dass der Nutzer aktiv etwas ausgelöst hat.	
📌 **3.2.2 Bei Eingabe [A]** *Wenn ein Nutzer eine Auswahl trifft oder etwas in ein Formular eingibt, darf sich die Seite nicht automatisch ändern, außer der Nutzer wurde vorher klar darauf hingewiesen.* 💡 Beispiel: Auf einer Bestellseite wählt man im Formular „Lieferung nach Hause“ aus. Dann erscheinen neue Felder für die Adresse – aber die Seite bleibt gleich und lädt nicht neu. Das ist okay, weil der Nutzer im selben Bereich weitermachen kann. 💡 Beispiel: Ein Nutzer tippt seine Telefonnummer ein. Sobald die Vorwahl eingegeben ist, springt der Cursor automatisch ins nächste Feld für die restliche Telefonnummer. Das ist erlaubt, wenn vorher ein kurzer Hinweis darübersteht, z. B. „Der Cursor springt automatisch weiter“.	
📌 **3.2.3 Einheitliche Navigation [AA]** *Wiederkehrende Navigationselemente und Steuerelemente sollten auf allen Seiten immer gleich aussehen und an der gleichen Stelle sein.* 💡 Beispiel: Die Hauptnavigation ist auf allen Unterseiten oben rechts und zeigt immer die gleichen Menüpunkte in der gleichen Reihenfolge – z. B. Startseite – Leistungen – Kontakt. 💡 Beispiel: Ein auffälliger Button „Jetzt bestellen“ ist auf jeder Produktseite unten rechts im sichtbaren Bereich platziert – so wissen Nutzer*innen immer, wo sie klicken müssen.	
📌 **3.2.4 Konsistente Identifizierung [AA]** *Wiederkehrende Funktionen müssen immer gleich benannt und dargestellt werden.* 💡 Beispiel: Auf allen Seiten heißt der Button für die Volltextsuche „Suchen“ – nicht mal „Finden“, mal „Go“ oder „Los“. 💡 Beispiel: Ein Symbol für das Herunterladen von PDFs wird auf allen Seiten mit „Download [Dateiname]“ als Alternativtext versehen. So wissen Screenreader-Nutzer*innen immer, was passiert.	

📌 **3.2.6 Konsistente Hilfe [A]** *Wenn eine Website Hilfe anbietet (z. B. Kontaktmöglichkeit, FAQs oder einen Chatbot), dann soll diese Hilfe auf allen Seiten an der gleichen Stelle zu finden sein.* *Dieses Kriterium gilt nur, wenn Hilfe tatsächlich vorhanden ist – es verpflichtet nicht dazu, Hilfe anzubieten.* 💡 Beispiel: Auf jeder Seite eines Bewerbungsportals ist oben rechts ein „Kontakt“-Link zu sehen, der auf ein Formular führt. 💡 Beispiel: In einem Online-Terminbuchungssystem ist auf jeder Seite unten ein Chatbutton zur Hilfe eingeblendet.	
📌 **3.3.1 Fehlererkennung [A]** *Fehler in Formularen müssen deutlich und verständlich angezeigt werden.* 💡 Beispiel: „E-Mail-Adresse fehlt“ unter dem Eingabefeld. 💡 Beispiel: Rotes Feld + Text: „Bitte geben Sie eine gültige PLZ ein“.	
📌 **3.3.2 Labels oder Anweisungen [A]** *Wenn Benutzer etwas eingeben sollen (z. B. in Formularfeldern), müssen passende Beschriftungen oder Anweisungen bereitgestellt werden, damit klar ist, was und wie etwas eingegeben werden soll.* 💡 Beispiel: Unter einem Feld steht: „Bitte im Format TT.MM.JJJJ eingeben“ → Das hilft, Eingabefehler zu vermeiden und ist eine klare Anweisung. 💡 Beispiel: Ein Textfeld für die E-Mail-Adresse ist mit „E-Mail-Adresse *“ beschriftet, wobei das Sternchen durch eine Anmerkung erklärt wird („* Pflichtfeld“). → Nutzer wissen sofort, dass die Angabe erforderlich ist.	
📌 **3.3.3 Vorschlag zur Fehlerbehebung [AA]** *Wenn ein Fehler bei der Eingabe erkannt wird (z. B. falsches Format oder fehlende Angaben), sollten dem Benutzer Korrekturvorschläge angezeigt werden, die ihm helfen, den Fehler zu beheben. Dies gilt nur, wenn die Korrektur den Sicherheitsaspekt oder den Zweck des Inhalts nicht gefährdet.* 💡 Beispiel: Ein Benutzer gibt „31.02.2025“ in ein Datumsfeld ein. Der Fehler wird erkannt, und der Benutzer bekommt einen Vorschlag wie: „Meinten Sie 28.02.2025?“. 💡 Beispiel: Ein Benutzer gibt „13“ in ein Feld für den Monat ein. Der Vorschlag zeigt: „Bitte einen der folgenden Monate eingeben: Januar, Februar, März…“.	

📌 **3.3.4 Fehlervermeidung (Recht, Finanzen, Daten) [AA]** *Benutzer haben vor wichtigen Aktionen (wie rechtlichen oder finanziellen Transaktionen) die Möglichkeit, Fehler zu überprüfen und zu korrigieren. Wenn zum Beispiel eine Bestellung oder eine wichtige Datenänderung stattfindet, sollten Benutzer die Möglichkeit haben, ihre Eingaben zu überprüfen, zu bestätigen oder zu ändern, bevor sie endgültig abgesendet werden.* 💡 Beispiel: Bevor der Benutzer eine Bestellung abschließt, zeigt der Online-Shop eine Übersicht der Bestellung an. 💡 Beispiel: Ein Benutzer möchte Aktien kaufen. Vor der endgültigen Bestätigung des Kaufs wird der Benutzer darauf hingewiesen, dass der Markt gerade geschlossen ist, und erhält die Möglichkeit, die Transaktion zu stornieren oder fortzufahren.	
📌 **3.3.7 Redundanter Eintrag [A]** *Benutzer müssen nicht dieselben Informationen mehrmals eingeben. Bereits eingegebene Daten sollten automatisch ausgefüllt oder zur Auswahl gestellt werden, um den Prozess zu erleichtern, besonders für Menschen mit Gedächtnisproblemen.* *Ausnahmen: Eine erneute Eingabe ist nur erforderlich, wenn: die Eingabe zwingend notwendig ist, die Information zur Sicherheit des Inhalts benötigt wird, oder die zuvor eingegebenen Daten nicht mehr gültig sind.* 💡 Beispiel: In einem Bestellprozess wird der Benutzer im ersten Schritt nach seiner Adresse gefragt. Im späteren Verlauf des Prozesses kann er einfach eine Option auswählen, die besagt: "Rechnungsadresse ist gleich Lieferadresse", und muss die Adresse nicht erneut eingeben. 💡 Beispiel: In einem Anmeldeformular für eine Veranstaltung wird der Benutzer gebeten, seine E-Mail-Adresse einzugeben. Wenn der Benutzer im nächsten Schritt ein weiteres Mal nach seiner E-Mail gefragt wird, könnte diese bereits automatisch vorausgefüllt oder dem Benutzer zur Auswahl gestellt werden, um den Aufwand zu minimieren.	
📌 **3.3.8 Zugängliche Authentifizierung (Minimum) [AA]** *Authentifizierungsprozesse dürfen keine kognitiven Tests (z. B. merken eines Passwortes oder Lösen eines Puzzles) erfordern.* *Wenn solche Tests notwendig sind, müssen sie durch alternative Methoden ersetzt oder unterstützt werden. Zum Beispiel durch Passwort-Manager, die Eingaben erleichtern, oder durch die Möglichkeit, Informationen zu kopieren und einzufügen.* *Es dürfen keine kognitiven Tests wie CAPTCHA erforderlich sein, es sei denn, sie erfüllen bestimmte Ausnahmen (zu. B. Barrierefreie Alternativen: Wenn ein CAPTCHA durch eine barrierefreie Lösung wie eine Audio-Version oder ein alternativer Test ersetzt wird.)* 💡 Beispiel: Barrierefreie CAPTCHA-Alternativen: Ein Online-Shop bietet ein Audio-Captcha an, das Menschen mit Sehbehinderungen ermöglicht, sich zu verifizieren, ohne ein visuelles CAPTCHA lösen zu müssen. 💡 Beispiel: Eine Bank verwendet ein CAPTCHA für die Anmeldung, erlaubt aber einen alternativen, barrierefreien Zugang durch den Kundendienst, der per Telefon den Zugang bestätigt, falls der Nutzer Probleme mit dem CAPTCHA hat.	

4. Robustheit

Der Inhalt muss robust genug sein, damit er von einer Vielzahl von Benutzeragenten, einschließlich unterstützender Technologien, interpretiert werden kann.

📌 **4.1.2 Name, Rolle, Wert [A]** *Stellen Sie sicher, dass alle Komponenten der Benutzeroberfläche (z. B. Formularelemente, Links) für Menschen, die assistierende Technologien nutzen, vollständig verständlich sind.* 💡 Beispiel: role="button" bei interaktiven Divs 💡 Beispiel: Wenn ein Akkordeon-Panel ein- oder ausgeklappt wird, kann das Attribut aria-expanded verwendet werden, um den aktuellen Zustand (offen oder geschlossen) zu kennzeichnen.	
📌 **4.1.3 Statusmeldungen [AA]** *Dynamische Inhalte, die den Benutzer über Änderungen informieren, müssen für Screenreader und unterstützende Technologien zugänglich gemacht werden, ohne dass der Fokus auf das Element gelegt wird.* 💡 Beispiel: Ein Screenreader gibt eine Meldung aus, wenn ein Benutzer eine Fehlermeldung beim Ausfüllen eines Formulars erhält: „Ungültige Eingabe in der Postleitzahl.“ 💡 Beispiel: Versandoption geändert → Meldung erscheint screenreaderfreundlich.	