

# Hydrophon Design-System

---

Dokumentation für Markenkonsistenz, Komponenten und technische Implementierung.

## Schnelleinstieg

Sie sind...	Starten Sie hier
Marketing / Agentur	<a href="#">Marketing Onboarding Guide</a>
Designer	<a href="#">Designer-Guide</a>
Entwickler	<a href="#">Entwickler-Guide</a>

---

## Für Marketing-Teams

Alles für Präsentationen, Social Media und Markenarbeit.

### Einstieg

- [Marketing Onboarding Guide](#) - Umfassender Einstieg (15-20 Seiten)

### Schnellzugriff

- Logo-Varianten und -Regeln
- Markenfarben
- Typografie

→ [Vollständiger Marketing-Guide](#)

---

## Für Designer

Komponenten, Patterns und Gestaltungsregeln.

### Design-Grundlagen

Thema	Dokumentation
Farben	<a href="#">Farbsystem</a>
Typografie	<a href="#">Schriftsystem</a>
Spacing & Grid	<a href="#">Abstands- und Rastersystem</a>
Effekte	<a href="#">Schatten &amp; Rundungen</a>
Icons	<a href="#">Icon-System</a>

## Komponenten

- Interaktiv: [Buttons](#)
- Formulare: [Index](#) | [Text Input](#) | [Validation](#)
- Navigation: [Index](#) | [Header](#) | [Mobile Drawer](#)
- Inhalte: [Product Card](#) | [Content Card](#) | [Data Table](#)
- Feedback: [Index](#) | [Modal](#) | [Toast](#)

## Markenidentität

- [Logo-Guidelines](#)
- [Vollständiger Designer-Guide](#)
- 

## Für Entwickler

Tokens, CSS-Variablen und technische Spezifikationen.

### Technische Referenz

- [Design Tokens Übersicht](#)
- [CSS-Variablen Referenz](#)
- [Grid & Breakpoints](#)

### Quick Start

```
@import 'design-system/build/css/variables.css';

.button {
  background: var(--button-primary-background-default);
}
```

## Komponenten mit Specs

Jede Komponenten-Dokumentation enthält technische Spezifikationen, ARIA-Patterns und Code-Beispiele:

- [Formulare](#) - Inputs, Validation, ARIA
  - [Navigation](#) - Focus Management, Keyboard Nav
  - [Feedback](#) - Live Regions, Modal Patterns
- [Vollständiger Entwickler-Guide](#)
- 

## Barrierefreiheit

WCAG 2.1 Level AA Compliance für alle Komponenten.

- [Accessibility Übersicht](#) - Grundprinzipien und ARIA-Patterns
- [WCAG Compliance](#) - Komponenten-spezifische Anforderungen
- [Testing Guide](#) - Prüfmethoden und Checklisten

## Dokumentationsstruktur

```
docs/
├── README.md                                # Diese Datei (Hub)
├── 00-quick-start/
|   └── marketing-onboarding.md            # Marketing Onboarding (15-20 S.)
├── buttons.md
├── colors.md
└── typography.md
    ├── spacing-grid.md
    ├── effects.md
    ├── icons.md
    └── logo-guidelines.md
    └── forms/
        ├── index.md
        ├── text-input.md
        ├── textarea.md
        ├── select.md
        ├── checkbox.md
        ├── radio-button.md
        ├── labels-helper-text.md
        └── validation.md
    └── navigation/
        ├── index.md
        ├── header.md
        ├── mobile-drawer.md
        ├── breadcrumb.md
        └── footer.md
    └── components/
        ├── product-card.md
        ├── content-card.md
        └── data-table.md
    └── feedback/
        ├── index.md
        ├── modal.md
        ├── tooltip.md
        ├── toast.md
        └── loading.md
    └── 03-accessibility/
        ├── overview.md
        ├── wcag-compliance.md
        └── testing-guide.md
    └── 04-technical-reference/
        ├── design-tokens.md
        ├── css-variables.md
        └── grid-breakpoints.md
    └── 05-audience-guides/
        ├── for-marketing.md
        ├── for-designers.md
        └── for-developers.md
```

## Asset-Dateien

Was	Pfad
Logo SVG	design-system/assets/logo/hydrophon/svg/
Logo PNG	design-system/assets/logo/hydrophon/png/
Token JSON	design-system/tokens/
CSS Variables	design-system/build/css/variables.css

Hydrophon Design-System v1.0 Dokumentation erstellt: Januar 2026

# Marketing Onboarding Guide – Hydrophon Design System

---

## Willkommen beim Hydrophon Design System

Dieser umfassende Leitfaden richtet sich an Marketing-Teams, externe Agenturen und alle, die mit der Hydrophon-Marke arbeiten, aber keine Designer sind. Sie lernen hier alles, was Sie benötigen, um on-brand Marketingmaterialien selbstständig und sicher zu erstellen.

**Ziel dieses Guides:** Nach der Lektüre können Sie eigenständig Präsentationen, Social Media Posts, Druckmaterialien und andere Marketing-Assets erstellen, die der Hydrophon-Markenidentität entsprechen.

---

## Inhaltsverzeichnis

1. Einleitung
  2. Logo-Verwendung
  3. Farbsystem
  4. Typografie-Grundlagen
  5. Asset-Zugang und Verwendung
  6. Häufige Marketing-Materialien
  7. Markenstimme und Ton
  8. Quick Reference Zusammenfassung
- 

## Einleitung

### Zweck dieses Guides

Hydrophon positioniert sich als moderne, innovative Marke im B2B-Sanitärbereich. Dieser Guide hilft Ihnen, diese Positionierung in allen Marketingmaterialien konsistent zu kommunizieren.

#### Was Sie hier finden:

- Klare Regeln für Logo, Farben und Typografie
- Praktische Beispiele für häufige Marketingmaterialien
- Do's and Don'ts mit konkreten Anwendungsfällen
- Wo Sie Assets finden und wie Sie sie verwenden

#### Was dieser Guide NICHT ist:

- Keine vollständige technische Entwickler-Dokumentation
- Kein tiefes Design-Handbuch für Designer
- Keine allgemeine Marketing-Strategie

## Wie man dieses Dokument nutzt

**Schneller Einstieg (2-5 Minuten):** Nutzen Sie die Quick Reference Tabellen in jedem Abschnitt für schnelle Antworten wie:

- Welches Logo auf dunklem Hintergrund?
- Was ist der Hex-Code für Hydrophon Blau?
- Welche Schriftart für Präsentationen?

**Gründliches Onboarding (30-60 Minuten):** Lesen Sie den gesamten Guide durch, um das Gesamtbild der Hydrophon-Marke zu verstehen und häufige Fehler zu vermeiden.

**Als Nachschlagewerk:** Verwenden Sie das Inhaltsverzeichnis, um gezielt Informationen zu finden, während Sie an spezifischen Projekten arbeiten.

## Wer sollte dieses Dokument lesen

**Primäre Zielgruppen:**

- **Marketing-Teams** – Erstellen von Kampagnen, Social Media, E-Mail-Marketing
- **Externe Agenturen** – Schnelles Onboarding in die Hydrophon-Marke
- **Vertriebsteams** – Erstellen von Präsentationen, Verkaufsunterlagen
- **Content-Erststeller** – Blog-Posts, Whitepapers, Case Studies

**Nicht primär für:**

- Entwickler (siehe technische Dokumentation)
- Designer (siehe vollständige Design-Guidelines)

**Bei tiefergehenden Fragen:** Wenden Sie sich an die Hydrophon Marketingabteilung.

## Logo-Verwendung

Das Hydrophon-Logo ist Ihr wichtigstes Markenelement. Korrekte Verwendung gewährleistet professionelle Markenpräsenz.

### Die 5 Logo-Varianten

Hydrophon bietet fünf Logo-Varianten für verschiedene Anwendungsfälle:

Variante	Aussehen	Wann verwenden
Original (Blau + Grau)	Zweifarbig: Blau #008BD2 + Grau #575656	Helle Hintergründe (Weiß, helles Grau) – <b>Standardvariante</b>
Weiß	Einfarbig weiß	Dunkle oder farbige Hintergründe
Schwarz	Einfarbig schwarz	Schwarz-Weiß-Druck (keine Farbe verfügbar)
Blau	Einfarbig blau	Einfarb-Druckverfahren, Gravur, Prägung
Grau	Einfarbig grau	Subtile Anwendungen (Wasserzeichen, Footer)

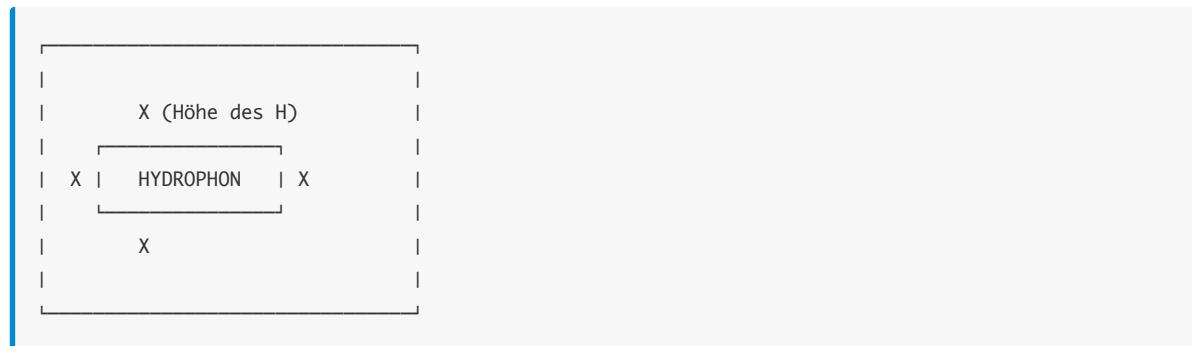
## Quick Reference: Welches Logo wann?

Situation	Logo-Variante	Mindestgröße	Hintergrund
Website-Header (heller Hintergrund)	Original	150px Breite	Weiß oder helles Grau
Website-Header (dunkler Hintergrund)	Weiβ	150px Breite	Dunkelblau, Dunkelgrau, Schwarz
PowerPoint-Präsentation (weiße Folie)	Original	150px Breite	Weiß
Social Media Post (farbiger Hintergrund)	Weiβ	180px Breite	Blau, Dunkel, Farbig
Schwarz-Weiß-Dokument	Schwarz	120px digital / 30mm Print	Weiß
E-Mail-Signatur	Original	120px Breite	Weiß
Print-Broschüre (Farbe)	Original	40mm Breite	Weiß oder hell
Visitenkarte	Original	30mm Breite	Weiß
Produkt-Gravur	Blau	30mm	Metall, Kunststoff

## Schutzraum (Clearspace)

**Regel:** Das Logo braucht Platz zum Atmen. Halten Sie mindestens **1X Abstand** auf allen Seiten frei.

**Was ist 1X?** Die Höhe des Buchstabens "H" in "Hydrophon".



**Wichtig:**

- Keine Texte im Schutzraum
- Keine anderen Logos im Schutzraum
- Keine Bilder oder Grafiken im Schutzraum
- Bei unruhigen Hintergründen (Fotos): **2X Schutzraum** verwenden

## Mindestgrößen

Digital (Website, E-Mail, Social Media):

- **Minimum:** 120px Breite
- **Empfohlen:** 150px Breite für optimale Lesbarkeit
- **Website-Header:** 200-250px Breite
- **Social Media Profilbild:** 180px Minimum

Print (Broschüren, Flyer, Visitenkarten):

- **Minimum:** 30mm Breite
- **Empfohlen:** 40mm Breite für optimale Lesbarkeit
- **Visitenkarten:** 35-40mm Breite

- **Briefpapier:** 40mm+ Breite

**Faustregel:** Wenn das Logo kleiner als die Mindestgröße sein muss, überdenken Sie die Platzierung. Ein zu kleines Logo ist unleserlich und schadet der Marke.

## Do's and Don'ts

- ✓ **DO – Richtig verwenden**
  - ✓ **Original-Logo auf hellem Hintergrund**
    - Nutzen Sie die Original-Variante (Blau + Grau) auf Weiß oder hellem Grau
  - ✓ **Weiße Logo auf dunklem Hintergrund**
    - Nutzen Sie die weiße Variante auf Dunkelblau, Dunkelgrau oder Schwarz
  - ✓ **Proportional skalieren**
    - Halten Sie immer die Proportionen des Logos
    - Ziehen Sie an den Ecken, nicht an den Seiten
  - ✓ **Ausreichend Kontrast**
    - Stellen Sie sicher, dass das Logo deutlich sichtbar ist
    - Mindestkontrast 4.5:1 (besser 7:1)
  - ✓ **Schutzraum einhalten**
    - Halten Sie 1X Abstand auf allen Seiten
    - Bei Fotos: 2X Schutzraum
- ✗ **DON'T – Nicht verwenden**
  - ✗ **Logo stauchen oder strecken**
    - Das Logo darf NIEMALS verzerrt werden
    - Immer nur proportional skalieren
  - ✗ **Logo rotieren oder schräg stellen**
    - Das Logo muss immer horizontal ausgerichtet sein
    - Keine Drehung um irgendeine Achse
  - ✗ **Schlagschatten oder Effekte hinzufügen**
    - Keine Schatten, Leuchteffekte, 3D-Effekte
    - Das Logo ist bereits final gestaltet
  - ✗ **Farben ändern**
    - Verwenden Sie nur die bereitgestellten Varianten
    - Keine individuellen Farbkombinationen
  - ✗ **Auf kontrastarmen Hintergründen**
    - Blaues Logo nicht auf blauem Hintergrund
    - Graues Logo nicht auf grauem Hintergrund
  - ✗ **Logo beschneiden**
    - Alle Teile des Logos müssen vollständig sichtbar sein
    - Schutzraum darf nicht abgeschnitten werden
  - ✗ **Zu nah an anderen Elementen**
    - Mindestabstand 1X einhalten
    - Bei komplexen Layouts: 2X Abstand

### **x Alte Logo-Versionen verwenden**

- Nutzen Sie nur die aktuellen offiziellen Dateien
- Bauen Sie das Logo nicht selbst nach

## Wo finde ich die Logo-Dateien?

Ordnerstruktur im Repository:

```
design-system/assets/logo/
└── hydrophon/
    ├── svg/                                ← Für Website, digitale Anwendungen
    │   ├── logo-hydrophon-original.svg
    │   ├── logo-hydrophon-weiss.svg
    │   ├── logo-hydrophon-schwarz.svg
    │   ├── logo-hydrophon-blau.svg
    │   └── logo-hydrophon-grau.svg
    ├── png/                                ← Für E-Mail, PowerPoint, Social Media
    │   ├── logo-hydrophon-original@2x.png
    │   ├── logo-hydrophon-weiss@2x.png
    │   └── ... (alle Varianten)
    └── favicon/                            ← Für Website-Favicons
        └── ... (verschiedene Größen)
└── gluy/                                ← Gluy Produktlinien-Logo
└── hyhero/                               ← hyHero Produktlinien-Logo
└── hyindustry/                           ← hyIndustry Produktlinien-Logo
```

### Welches Format wann?

- **SVG:** Website, Web-Apps, moderne digitale Anwendungen (bevorzugt)
- **PNG @2x:** E-Mail-Signaturen, PowerPoint, Social Media, Legacy-Systeme
- **PNG @3x:** High-Resolution-Displays (iPhone, moderne Tablets)

Für Print-Materialien: Kontaktieren Sie die Marketingabteilung für hochauflösende PDF-CMYK-Dateien.

Mehr Details: [Logo-Guidelines](#) (vollständige technische Dokumentation)

## Farbsystem

Farben sind ein zentrales Element der Hydrophon-Marke. Richtiger Farbeinsatz stärkt Wiedererkennung und Professionalität.

### Die 2 Primärfarben

Hydrophon hat zwei Hauptfarben, die in allen Materialien verwendet werden:

Farbe	Hex-Code	RGB	Wann verwenden
Hydrophon Blau	#008BD2	0, 139, 210	Call-to-Actions, Links, Buttons, Markenakzente
Hydrophon Grau	#575656	87, 86, 86	Texte, Icons, sekundäre Elemente

Visualisierung:

Hydrophon Blau – #008BD2

  
 Ihr primäres Markenelement  
 Verwendung: Buttons, Links, Akzente  


#### Hydrophon Grau – #575656

  
 Für Texte und Icons  
 Verwendung: Überschriften, Fließtext, Icons  


### Quick Reference: Farbanwendung

Anwendung	Farbe	Hex-Code
Primärer Call-to-Action Button	Hydrophon Blau	#008BD2
Links im Text	Hydrophon Blau	#008BD2
Hauptüberschriften	Hydrophon Grau	#575656
Fließtext	Dunkelgrau	#171717
Sekundärer Button (z.B. "Abbrechen")	Hydrophon Grau	#575656
Icons (nicht klickbar)	Hydrophon Grau	#575656
Icons (klickbar)	Hydrophon Blau	#008BD2
Haupthintergrund	Weiβ	#FFFFFF
Sekundärer Hintergrund	Hellgrau	#FAFAFA

### Produktlinien-Farben

Jede Hydrophon-Produktlinie hat eigene Farben für produktspezifische Kommunikation:

Produktlinie	Primärfarbe	Hex-Code	Wann verwenden
hyHero	Orange	#F49A36	hyHero-Produktseiten, Datenblätter, Social Posts zu hyHero
hyIndustry	Dunkelblau	#0E2638	hyIndustry-Produktseiten, Datenblätter, Social Posts zu hyIndustry
Gluy	Gelb	#FFEEB6	Gluy-Produktseiten, Datenblätter, Social Posts zu Gluy

**Wichtig:** Produktlinien-Farben sind **Akzentfarben**, keine Ersatz für Hydrophon Blau und Grau.

### Funktionale Farben

Für Status-Mitteilungen und System-Feedback verwenden Sie diese Farben:

Funktion	Farbe	Hex-Code	Verwendung
Erfolg	Grün	#22C55E	"Formular erfolgreich gesendet", Bestätigungen
Warnung	Orange	#F59E0B	"Bitte prüfen Sie Ihre Eingabe", wichtige Hinweise
Fehler	Rot	#EF4444	"Fehler aufgetreten", Validierungsfehler, Löschen-Button
Information	Hellblau	#3B82F6	Neutrale Infos, Tooltips, "Neue Funktionen verfügbar"

**Wichtig:** Verwenden Sie funktionale Farben konsistent:

- Erfolg ist IMMER grün
- Fehler ist IMMER rot
- Warnung ist IMMER orange
- Verwechseln Sie diese nie mit Markenfarben

## Do's and Don'ts – Farben

### ✓ DO – Richtige Farbverwendung

#### ✓ Hydrophon Blau für primäre Aktionen

- Call-to-Action-Buttons
- Wichtige Links
- Primäre Icons (klickbar)

#### ✓ Hydrophon Grau für Texte

- Überschriften
- Fließtext (oder dunkleres Grau #171717 )
- Sekundäre UI-Elemente

#### ✓ Ausreichend Kontrast

- Text muss lesbar sein (4.5:1 Minimum)
- Weißer Text auf Hydrophon Blau ✓
- Dunkelgrauer Text auf Weiß ✓

#### ✓ Produktlinien-Farben als Akzente

- hyHero-Produktseite: Orange-Akkzente + Hydrophon Blau/Grau
- Gluy-Social-Post: Gelbe Elemente + Hydrophon Blau/Grau

### ✗ DON'T – Fehlerhafte Farbverwendung

#### ✗ Produktlinien-Farben als Hauptfarben

- hyHero Orange NICHT für alle Buttons verwenden
- Gluy Gelb NICHT als Hauphintergrund verwenden
- Produktlinien-Farben sind Akzente, keine Ersatzfarben

#### ✗ Funktionale Farben verwechseln

- Erfolg nicht in Rot darstellen
- Fehler nicht in Grün darstellen
- Hydrophon Blau nicht für Fehlermeldungen

#### ✗ Unzureichender Kontrast

- Hellgraue Texte auf weißem Hintergrund
- Gelber Text auf weißem Hintergrund
- Prüfen Sie Kontrast mit Tools (WebAIM Contrast Checker)

- ✗ **Produktlinien-Farben mischen**
  - hyHero Orange + Guy Gelb zusammen verwenden
  - Jede Produktkommunikation nutzt nur ihre eigene Farbe
- ✗ **Zu viele Farben**
  - Mehr als 3-4 Farben pro Design verwirren
  - Fokus auf Hydrophon Blau und Grau

## Praktische Beispiele

### Beispiel 1: Website-Button

- Hintergrund: Hydrophon Blau ( #008BD2 )
- Text: Weiß ( #FFFFFF )
- Hover: Dunkleres Blau ( #007BB8 )

### Beispiel 2: Präsentations-Folie

- Hintergrund: Weiß ( #FFFFFF )
- Überschrift: Hydrophon Grau ( #575656 )
- Akzent-Element: Hydrophon Blau ( #008BD2 )

### Beispiel 3: Social Media Post zu hyHero

- Hintergrund: Weiß oder hyHero Orange-Akzente
- Text: Hydrophon Grau
- Button: Hydrophon Blau
- Produktnamen: hyHero Orange ( #F49A36 )

**Mehr Details:** [Farbsystem](#) (vollständige Farbpalette mit allen Nuancen)

---

## Typografie-Grundlagen

Die richtige Schrift und Schriftgrößen sorgen für professionelle, lesbare Kommunikation.

### Die Primärschrift: Inter

Hydrophon verwendet Inter für ALLE Texte (Überschriften, Fließtext, UI-Elemente).

#### Warum Inter?

- Modern und professionell
- Exzellente Lesbarkeit auf Bildschirmen
- Kostenlos verfügbar (Google Fonts)
- Vermittelt Innovation und technische Glaubwürdigkeit

#### Wo herunterladen:

- Google Fonts: <https://fonts.google.com/specimen/Inter>
- Benötigte Schriftstärken:
  - Regular (400) – Für Fließtext
  - Medium (500) – Für Labels, UI
  - Semibold (600) – Für Subheadings
  - Bold (700) – Für Hauptüberschriften

## Schriftgrößen für Marketing-Materialien

### Präsentationen (PowerPoint, Keynote, Google Slides)

Element	Schriftgröße	Schriftstärke	Verwendung
Folientitel	32-36px	Bold (700)	Jede Folie hat einen Titel
Hauptüberschrift (erste Folie)	48px	Bold (700)	Präsentationstitel, Hero-Slide
Unterüberschriften	24-28px	Semibold (600)	Abschnitte innerhalb Folie
Fließtext	18-20px	Regular (400)	Niemals kleiner als 18px!
Kleintext (Fußnoten)	14px	Regular (400)	Quellen, rechtliche Hinweise

#### Wichtig für Präsentationen:

- Niemals unter 18px für Haupttext (Raum ist oft groß, Publikum weit weg)
- Headlines 32px+ für gute Lesbarkeit
- Maximale Textmenge: 6-8 Zeilen pro Folie

### Social Media Posts

Plattform	Empfohlene Schriftgröße	Hinweis
Instagram Post	24-32px	Große Schrift für mobile Lesbarkeit
Instagram Story	28-36px	Noch größer, schnelle Lesbarkeit erforderlich
LinkedIn Post	20-28px	Moderate Größe, oft auf Desktop gesehen
Facebook Post	20-28px	Wie LinkedIn
Twitter/X	18-24px	Kompaktere Schrift akzeptabel

Allgemeine Regel: Mobile-first denken. Schrift muss auf Smartphone-Bildschirmen lesbar sein.

### Print-Materialien (Flyer, Broschüren, Poster)

Element	Schriftgröße	Schriftstärke	Verwendung
Hauptüberschrift	36-48pt	Bold (700)	Titel auf Titelseite
Abschnittsüberschriften	24-30pt	Semibold (600)	Kapitelüberschriften
Fließtext	10-12pt	Regular (400)	Standard-Leseschriftgröße
Bildunterschriften	8-9pt	Regular (400)	Captions, Metadaten

Wichtig: Print-Größen in pt (Points), nicht px.

### E-Mail-Signaturen

Element	Schriftgröße	Schriftstärke
Name	14-16px	Semibold (600)
Position	12-14px	Regular (400)
Kontaktdaten	11-12px	Regular (400)

## Überschriften vs. Fließtext

### Überschriften (Headings):

- **H1 (Hauptüberschrift):** 48px, Bold – Eine pro Seite/Folie
- **H2 (Abschnitt):** 36px, Bold – Hauptabschnitte
- **H3 (Unterabschnitt):** 30px, Semibold – Unterabschnitte
- **H4 (Card-Titel):** 24px, Semibold – Kleinere Überschriften

### Fließtext (Body):

- **Standard:** 16px, Regular – Alle Fließtexte
- **Groß (Lead):** 18px, Regular – Einleitungsabsätze
- **Klein (Captions):** 12-14px, Regular – Metadaten, Fußnoten

**Faustregel:** Je größer der Text, desto fetter die Schriftstärke.

## Do's and Don'ts – Typografie

### ✓ DO – Richtige Schriftverwendung

#### ✓ Inter für alles

- Konsistent Inter für Überschriften und Fließtext verwenden
- Keine anderen Schriftarten mischen

#### ✓ Lesbare Größen

- Präsentationen: Mindestens 18px für Fließtext
- Print: Mindestens 10pt für Fließtext
- Social Media: Mindestens 20px für mobile Lesbarkeit

#### ✓ Klare Hierarchie

- Große, fette Überschriften
- Kleinere, normale Fließtexte
- Deutlicher Unterschied zwischen H1, H2, H3

#### ✓ Ausreichend Zeilenabstand

- Fließtext: 1.5x Zeilenhöhe (z.B. 16px Text = 24px Zeilenhöhe)
- Überschriften: 1.2x Zeilenhöhe

### ✗ DON'T – Fehlerhafte Schriftverwendung

#### ✗ Comic Sans oder andere unpassende Schriften

- Keine dekorativen Schriften
- Keine Script-Schriften (Handschrift-Stil)
- Bleiben Sie bei Inter

#### ✗ Zu kleine Schrift

- Präsentationen NICHT unter 18px für Haupttext
- Print NICHT unter 10pt für Fließtext
- Social Media NICHT unter 20px

#### ✗ Zu viele Schriftstärken

- Maximal 3 Schriftstärken pro Design
- Standard: Regular, Semibold, Bold

#### ✗ Alles in Großbuchstaben (ALL CAPS)

- Nur für kurze Headlines oder Buttons

- Fließtext NIEMALS in ALL CAPS (unleserlich)

**x Zu lange Zeilen**

- Maximal 75 Zeichen pro Zeile für Lesbarkeit
- Bei breiten Layouts: Text in Spalten aufteilen

## Praktische Beispiele

### Beispiel 1: PowerPoint-Folie

[Folientitel]  
Hydrophon AquaTech Serie  
Schrift: Inter Bold, 36px, Hydrophon Grau

[Fließtext]  
Modernste Technologie für gewerbliche  
Sanitäranlagen. Effizient, zuverlässig, innovativ.  
Schrift: Inter Regular, 20px, Dunkelgrau

### Beispiel 2: Social Media Post (Instagram)

[Headline]  
Neu: AquaTech Pro 3000  
Schrift: Inter Bold, 32px, Hydrophon Blau

[Body]  
Die nächste Generation ist da  
Schrift: Inter Regular, 24px, Hydrophon Grau

### Beispiel 3: Broschüren-Titelseite

[Titel]  
Produktkatalog 2026  
Schrift: Inter Bold, 48pt, Hydrophon Grau

[Untertitel]  
Innovative Lösungen für den Sanitärbereich  
Schrift: Inter Regular, 18pt, Hydrophon Grau

Mehr Details: [Typografie-Guidelines](#) (vollständige Schriftspezifikationen)

## Asset-Zugang und Verwendung

Wo finden Sie alle Marken-Assets und wie verwenden Sie sie richtig?

### Ordnerstruktur

Alle Hydrophon Design-Assets befinden sich im zentralen Repository:

```

design-system/
├── assets/
|   └── logo/
|       ├── hydrophon/           ← Hauptmarke
|       ├── gluy/                ← Gluy-Produktlinie
|       ├── hyhero/              ← hyHero-Produktlinie
|       └── hyindustry/          ← hyIndustry-Produktlinie
└── docs/
    ├── logo-guidelines.md     ← Detaillierte Logo-Regeln
    ├── colors.md              ← Vollständige Farbpalette
    └── typography.md          ← Typografie-Spezifikationen
    └── tokens/
        └── ...                  ← Für Entwickler (ignorieren Sie diese)

```

#### Wo Sie Zugang erhalten:

1. **Internes Team:** Zugriff via Unternehmens-Repository (Git)
2. **Externe Agenturen:** Kontakt Marketingabteilung für Asset-Paket
3. **Einzelne Assets:** Via Marketingabteilung anfragen

#### Dateiformate: Wann welches?

Format	Verwendung	Wann nutzen
SVG	Website, Web-Apps	Bevorzugt für alle digitalen Anwendungen – skaliert verlustfrei
PNG @2x	E-Mail, PowerPoint, Social Media	Standard für Raster-Grafiken – breite Kompatibilität
PNG @3x	iPhone, High-Res-Displays	Nur wenn explizit höhere Auflösung benötigt
PDF (CMYK)	Professioneller Druck	Kontakt Marketingabteilung – spezielle Druckdateien

#### Einfache Regel:

- **Website:** SVG verwenden
- **Präsentationen/E-Mail:** PNG @2x verwenden
- **Print:** Marketingabteilung kontaktieren

#### Namenskonventionen

Alle Dateien folgen einem einheitlichen Namensschema (lowercase mit Bindestrichen):

#### Struktur:

```
logo-[produktlinie]-[variante].[format]
```

#### Beispiele:

- logo-hydrophon-original.svg
- logo-hydrophon-weiss@2x.png
- logo-gluy-original.svg
- logo-hyhero-schwarz@2x.png

#### Produktlinien:

- hydrophon – Hauptmarke
- gluy , hyhero , hyindustry – Produktlinien

#### Varianten:

- `original` – Standardversion (mehrfarbig)
- `weiss`, `schwarz`, `blau`, `grau` – Einfarbige Versionen

### Wie verwende ich Assets in verschiedenen Programmen?

#### PowerPoint / Keynote / Google Slides

1. Öffnen Sie den Ordner `design-system/assets/logo/hydrophon/png/`
2. Wählen Sie die passende Variante (z.B. `logo-hydrophon-original@2x.png`)
3. Einfügen → Bild → Aus Datei
4. Größe anpassen (mindestens 150px Breite)
5. Schutzraum beachten (siehe Logo-Abschnitt)

#### E-Mail-Signatur

1. Verwenden Sie `logo-hydrophon-original@2x.png`
2. Größe: 120-150px Breite
3. Fügen Sie als Bild (nicht als Anhang) ein
4. Verlinken Sie optional mit der Hydrophon-Website

#### Social Media Posts

1. Verwenden Sie PNG @2x oder @3x für hohe Qualität
2. Instagram: Mindestens 180px Breite
3. LinkedIn/Facebook: Mindestens 150px Breite
4. Beachten Sie Schutzraum auf farbigen Hintergründen

#### Adobe Photoshop / Illustrator / Figma

1. Für Vektorgrafik-Programme: SVG verwenden (skaliert verlustfrei)
2. Für Photoshop: PNG @2x verwenden
3. Behalten Sie das Vektorformat bei (nicht in Pixel umwandeln)

### Wo finde ich zusätzliche Materialien?

#### Design-Assets verfügbar:

- ✓ Logos (alle Varianten, alle Produktlinien)
- ✓ Favicons (Website-Icons, App-Icons)
- ✓ Farbpalette (vollständige Dokumentation)
- ✓ Typografie-Spezifikationen

#### Noch nicht verfügbar / auf Anfrage:

- Produktfotos → Produktmanagement
- Stock-Fotos für Kampagnen → Marketingabteilung
- Templates (PowerPoint, Flyer) → In Entwicklung, Marketingabteilung fragen
- Video-Assets → Video-Team

### Wie fordere ich neue Assets an?

Kontakt: Hydrophon Marketingabteilung

Bei Anfrage angeben:

1. Was benötigen Sie? (z.B. "Logo für Messestand", "Print-PDF für Broschüre")

2. In welchem Format? (SVG, PNG, PDF)
3. Für welchen Zweck? (Website, Print, Präsentation)
4. Deadline? (Wann brauchen Sie es)

**Typische Anfragen:**

- "Ich brauche das Logo in Gold für eine Prägung" → Spezielle Variante
  - "Ich brauche hochauflösende PDFs für Druckerei" → PDF-CMYK-Dateien
  - "Ich brauche das Logo für T-Shirt-Druck" → Spezielle Druckdatei
- 

## Häufige Marketing-Materialien

Wie erstellen Sie on-brand Materialien für die häufigsten Anwendungsfälle?

### Präsentationen (PowerPoint, Keynote, Google Slides)

#### Layout-Tipps

**Folientitel:**

- Position: Oben links oder zentriert
- Schrift: Inter Bold, 32-36px, Hydrophon Grau
- Abstand zum oberen Rand: 40-60px

**Fließtext:**

- Position: Unter Titel, linksbündig
- Schrift: Inter Regular, 18-20px, Dunkelgrau
- Zeilenhöhe: 1.5x (z.B. 20px Text = 30px Zeilenhöhe)
- Maximale Textmenge: 6-8 Zeilen pro Folie

**Logo:**

- Position: Oben links oder unten rechts (konsistent über alle Folien)
- Größe: 150-200px Breite
- Variante: Original (auf weißen Folien) oder Weiß (auf dunklen Folien)

### Farbverwendung in Präsentationen

**Hintergrund:**

- Standard: Weiß ( #FFFFFF )
- Alternative: Hydrophon Blau ( #008BD2 ) für Titelfolien oder Akzente
- Vermeiden: Zu viele verschiedene Hintergrundfarben (maximal 2 pro Präsentation)

**Akzentelemente:**

- Hydrophon Blau für Buttons, Icons, Hervorhebungen
- Hydrophon Grau für Trennlinien, subtile Elemente
- Produktlinien-Farben nur bei produktspezifischen Folien

## Beispiel: Standard-Folie

[Logo]	
150px	
Folientitel	
Inter Bold, 36px, Hydrophon Grau	
• Bullet Point 1	
• Bullet Point 2	
• Bullet Point 3	
Inter Regular, 20px, Dunkelgrau	

## Präsentations-Checkliste

- Logo auf jeder Folie (konsistente Position)
- Schriftgröße mindestens 18px für Fließtext
- Titel 32px+, fett
- Maximal 6-8 Zeilen Text pro Folie
- Hydrophon Blau für primäre Akzente
- Konsistente Farben über alle Folien

## Social Media Posts

### Instagram

#### Bildgröße:

- Feed Post: 1080×1080px (quadratisch) oder 1080×1350px (vertikal)
- Story: 1080×1920px (vertikal)

#### Text im Bild:

- Schrift: Inter Bold, 28-36px für mobile Lesbarkeit
- Maximale Textlänge: 2-3 kurze Zeilen
- Farbe: Hydrophon Blau oder Weiß (auf Foto)

#### Logo:

- Position: Unten rechts oder oben links
- Größe: 180-200px
- Variante: Weiß (auf Fotos/farbigen Hintergründen) oder Original (auf Weiß)

#### Farbpalette:

- Hintergrund: Weiß, Hydrophon Blau, oder Produktlinien-Farbe (für Produktposts)
- Text: Kontrastreicher Text (Weiß auf Dunkel, Dunkel auf Hell)

### LinkedIn / Facebook

#### Bildgröße:

- Post: 1200×630px (Querformat)

#### Text im Bild:

- Schrift: Inter Bold, 24-32px
- Farbe: Hydrophon Grau oder Weiß

#### Logo:

- Position: Unten rechts
- Größe: 150px
- Variante: Passend zum Hintergrund

#### Social Media Checkliste

- Logo sichtbar und lesbar (mindestens 150px)
- Schriftgröße 24px+ für mobile Geräte
- Hydrophon-Farben dominant (Blau und/oder Grau)
- Produktlinien-Farben nur bei Produktposts
- Text kurz und knackig (maximal 3 Zeilen im Bild)
- Kontrast: Lesbar auf allen Geräten

#### Druckmaterialien (Flyer, Broschüren, Plakate)

##### CMYK vs. RGB

**Wichtig:** Print verwendet CMYK-Farbraum, nicht RGB.

##### Kontaktieren Sie die Marketingabteilung für:

- CMYK-Farbwerte von Hydrophon Blau und Grau
- Hochauflösende Logo-PDFs für Druck
- Druckvorlagen mit korrektem Beschnitt

**Nicht selbst umrechnen:** RGB zu CMYK Konvertierung variiert je nach Druckverfahren.

##### Mindestgrößen für Print

#### Logo:

- Mindestens 30mm Breite
- Empfohlen 40mm+ für Broschüren
- Größere Formate (Plakate): Nach Proportionen skalieren

#### Schrift:

- Fließtext: Mindestens 10pt
- Überschriften: Mindestens 18pt
- Kleintext (Fußnoten): Mindestens 8pt

#### Beschnitt (Bleed)

**Was ist Beschnitt?** Extra 3-5mm um das Design herum, damit beim Schneiden keine weißen Ränder entstehen.

**Regel:** Platzieren Sie wichtige Elemente (Text, Logo) mindestens 5mm vom Schnittrand entfernt.

**Kontakt Druckerei:** Fragen Sie nach Beschnitt-Anforderungen (meist 3mm Standard).

#### Print-Checkliste

- CMYK-Farben von Marketingabteilung erhalten
- Logo mindestens 30mm Breite
- Schriftgröße mindestens 10pt für Fließtext
- Beschnitt (Bleed) 3-5mm beachtet
- Wichtige Elemente 5mm+ vom Rand

- Druckauflösung mindestens 300 DPI

## E-Mail-Marketing

### E-Mail-Header

#### Logo:

- Größe: 120-150px Breite
- Position: Oben links oder zentriert
- Format: PNG @2x (beste Kompatibilität)

#### Hintergrund:

- Standard: Weiß
- Alternative: Hydrophon Blau (dann weißes Logo verwenden)

### Call-to-Action Buttons

#### Design:

- Hintergrund: Hydrophon Blau ( #008BD2 )
- Text: Weiß, Inter Medium, 14-16px
- Padding: 12px 24px (Button-Höhe ~40-48px)
- Border-Radius: 4px (leicht abgerundete Ecken)

#### Text:

- Kurz und actionable: "Jetzt anfragen", "Mehr erfahren", "Kontakt aufnehmen"
- Keine langen Sätze

### Text-Formatierung

#### Überschrift:

- Inter Semibold, 20-24px, Hydrophon Grau

#### Fließtext:

- Inter Regular, 14-16px, Dunkelgrau
- Zeilenhöhe: 1.5x

#### Links:

- Farbe: Hydrophon Blau
- Unterstrichen oder fett (deutlich erkennbar)

### E-Mail-Checkliste

- Logo 120-150px Breite im Header
- CTA-Button in Hydrophon Blau
- Schriftgröße mindestens 14px
- Links deutlich erkennbar (Blau, unterstrichen)
- Mobile-optimiert (Text lesbar auf Smartphone)

## Visitenkarten

### Logo:

- Größe: 30-40mm Breite
- Position: Vorderseite oben links oder zentriert
- Variante: Original (auf Weiß) oder Weiß (auf farbigem Hintergrund)

**Text:**

- Name: Inter Semibold, 12-14pt
- Position: Inter Regular, 10-11pt
- Kontaktdaten: Inter Regular, 9-10pt

**Farben:**

- Vorderseite: Weiß mit Hydrophon Blau/Grau Elementen
  - Rückseite: Optional Hydrophon Blau (dann weißer Text/Logo)
- 

## Markenstimme und Ton

Wie kommuniziert Hydrophon? Welche Sprache, welcher Ton, welche Botschaft?

### Hydrophons Positionierung

**Kernbotschaft:** Hydrophon ist ein **moderner, innovativer Anbieter** im B2B-Sanitärbereich, der technische Exzellenz mit zukunftsorientiertem Denken verbindet.

**Wofür steht Hydrophon?**

- **Innovation:** Wir denken voraus, entwickeln neue Lösungen
- **Qualität:** Zuverlässig, langlebig, professionell
- **Expertise:** Jahrelange Erfahrung im Sanitärbereich
- **Zukunftsorientiert:** Nachhaltig, digital, modern

**Was Hydrophon NICHT ist:**

- Nicht almodisch oder traditionsverhaftet
- Nicht billig oder Low-Budget
- Nicht kompliziert oder unnahbar
- Nicht übertrieben werblich oder marktschreierisch

### Kommunikationsstil

**Tonalität: Sachlich, aber zugänglich**

**Richtig:**

- "Die AquaTech Serie vereint modernste Technologie mit bewährter Qualität."
- "Unsere Lösungen sind für höchste Ansprüche im gewerblichen Bereich entwickelt."
- "Kontaktieren Sie uns für eine individuelle Beratung."

**Falsch:**

- "Die ULTIMATIVE Lösung für ALLE Ihre Probleme!!!" (zu werblich)
- "Gemäß §12 Absatz 3..." (zu formell, zu juristisch)
- "Hey, check mal unsere coolen Produkte!" (zu informell)

**Ansprache: Sie-Form, professionell**

**Immer "Sie":**

- "Haben Sie Fragen? Wir beraten Sie gerne."
- "Entdecken Sie unsere Produktpalette."

**Niemals "Du":**

- "Hast Du Fragen?" **✗**

- "Entdecke unsere Produkte!" 

**Ausnahme:** Social Media kann gelegentlich informeller sein, bleibt aber professionell.

### Sprache: Klar, verständlich, technisch kompetent

Fachbegriffe verwenden, aber erklären:

- "Die integrierte Steuerungseinheit ermöglicht präzise Regelung der Durchflussmenge." ✓
- "Unser patentiertes Hydro-Flux-System optimiert die Wasserzirkulation." ✓

Vermeiden Sie:

- Zu viel Fachjargon ohne Kontext
- Anglizismen, wenn deutsche Begriffe existieren
- Marketingfloskeln ohne Substanz ("revolutionär", "einzigartig", "unschlagbar")

### Beispiele: Gute vs. schlechte Formulierungen

Situation	 Schlecht	✓ Gut
Produktbeschreibung	"Das BESTE Sanitärmoodul ever!"	"Hochleistungs-Sanitärmoodul mit integrierter Steuerung für gewerbliche Anwendungen."
Call-to-Action	"Jetzt zuschlagen!!!!"	"Jetzt beraten lassen" oder "Kontakt aufnehmen"
Fehlerseite	"Oops, da ist was schiefgelaufen!"	"Die Seite konnte nicht geladen werden. Bitte versuchen Sie es erneut."
E-Mail-Betreff	"🔥 HOT DEAL: 50% RABATT!!!"	"Neue AquaTech Serie verfügbar – Jetzt informieren"
Social Media	"OMG ihr müsst das sehen 😱 😱 😱"	"Neues aus der Produktentwicklung: AquaTech Pro 3000 ist da."

### Was Sie vermeiden sollten

#### Altmodische Floskeln

##### Zu vermeiden:

- "Hochachtungsvoll"
- "In der Anlage erhalten Sie..."
- "Wir möchten Ihnen mitteilen, dass..."
- "Diesbezüglich..."

##### ✓ Besser:

- "Mit freundlichen Grüßen" oder "Beste Grüße"
- "Hier finden Sie..."
- "Wir freuen uns, Ihnen mitzuteilen..." oder einfach die Info
- "Hierzu..."

### Übertriebene Werbesprache

##### Zu werblich:

- "Einmaliges Angebot!"
- "Jetzt oder nie!"
- "Das Beste auf dem Markt!"
- "Revolutionäre Innovation!"

✓ **Besser:**

- "Zeitlich begrenztes Angebot"
- "Verfügbar bis [Datum]"
- "Hochwertige Lösung für..."
- "Innovative Technologie"

**Umgangssprache**

✗ **Zu informell:**

- "Krasse Sache!"
- "Voll gut"
- "Check das mal aus"
- "Mega cool"

✓ **Besser:**

- "Beeindruckende Lösung"
- "Hervorragende Qualität"
- "Entdecken Sie"
- "Innovatives Design"

**Praktische Beispiele nach Medium**

**E-Mail-Text (an Kunden)**

**Betreff:** Neue Produktserie AquaTech – Jetzt informieren

**Body:**

Sehr geehrte/r [Name],  
wir freuen uns, Ihnen unsere neue AquaTech Serie vorzustellen. Die Serie vereint modernste Technologie  
**Highlights:**

- Integrierte Steuerungseinheit für präzise Regelung
- Energieeffiziente Bauweise
- Modulare Erweiterbarkeit

Gerne beraten wir Sie persönlich zu den neuen Möglichkeiten für Ihr Projekt.  
Beste Grüße  
Ihr Hydrophon Team

**Tonalität:** Professionell, informativ, einladend – nicht übertrieben werblich.

**Social Media Post (LinkedIn)**

**Post-Text:**

Neues aus der Produktentwicklung: Die AquaTech Pro 3000 ist ab sofort verfügbar.

Mit integrierter Steuerungseinheit und energieeffizienter Bauweise setzt die AquaTech Serie neue Maßstäbe.

Mehr Informationen: [Link]

#Hydrophon #Sanitärtechnik #Innovation

**Tonalität:** Sachlich, kompetent, keine Emojis oder Umgangssprache.

#### Website-Hero-Text (Startseite)

**Headline:** Innovative Lösungen für den Sanitärbereich

**Subheadline:** Hydrophon entwickelt hochwertige Sanitärmodule für gewerbliche Anwendungen – zuverlässig, effizient und zukunftsorientiert.

**CTA-Button:** Produkte entdecken

**Tonalität:** Klar, prägnant, wertorientiert.

#### Marken-Do's and Don'ts Zusammenfassung

##### ✓ DO – Richtige Kommunikation

✓ Sie-Form verwenden ✓ Sachlich und professionell ✓ Technische Kompetenz zeigen ✓ Klare, verständliche Sprache ✓ Wertorientiert kommunizieren (Qualität, Innovation, Expertise) ✓ Call-to-Actions klar formulieren ("Jetzt beraten lassen", "Kontakt aufnehmen")

##### ✗ DON'T – Fehlerhafte Kommunikation

✗ Du-Form (außer explizit genehmigt) ✗ Übertriebene Werbesprache ("MEGA DEAL!!!") ✗ Umgangssprache ("Krass", "Voll gut") ✗ Altmodische Floskeln ("Hochachtungsvoll") ✗ Zu technisch ohne Kontext (Fachjargon-Überladung) ✗ Substanzlose Marketing-Floskeln ("revolutionär", "einzigartig", ohne Begründung)

#### Quick Reference Zusammenfassung

Die 10 wichtigsten Regeln auf einen Blick:

##### 1. Logo

✓ Original-Variante (Blau + Grau) auf hellem Hintergrund ✓ Weiße Variante auf dunklem Hintergrund ✓ Mindestgröße 120px (digital) / 30mm (print) ✓ Schutzraum 1X auf allen Seiten ✗ Niemals stauchen, strecken, rotieren oder Effekte hinzufügen

##### 2. Farben

✓ Hydrophon Blau (#008BD2) für primäre Aktionen, Buttons, Links ✓ Hydrophon Grau (#575656) für Texte, Icons ✓ Produktlinien-Farben nur als Akzente bei produktsspezifischer Kommunikation ✗ Produktlinien-Farben nicht als Hauptfarben verwenden ✗ Funktionale Farben nicht verwechseln (Erfolg = Grün, Fehler = Rot)

### 3. Typografie

- ✓ Inter für alle Texte ✓ Präsentationen: Mindestens 18px Fließtext, 32px+ Headlines ✓ Social Media: Mindestens 20-24px für mobile Lesbarkeit ✓ Print: Mindestens 10pt Fließtext ✗ Niemals unter 18px bei Präsentationen ✗ Keine anderen Schriftarten mischen

### 4. Assets finden

- ✓ SVG für Website und digitale Anwendungen ✓ PNG @2x für E-Mail, PowerPoint, Social Media ✓ Marketingabteilung kontaktieren für Print-PDFs (CMYK)

### 5. Präsentationen

- ✓ Logo auf jeder Folie (konsistente Position, 150-200px) ✓ Folientitel 32-36px, fett ✓ Fließtext mindestens 18px ✓ Maximal 6-8 Zeilen pro Folie ✓ Hydrophon Blau für Akzente

### 6. Social Media

- ✓ Logo mindestens 150-180px ✓ Schriftgröße mindestens 24px für mobile Lesbarkeit ✓ Hydrophon-Farben dominant ✓ Text kurz (maximal 3 Zeilen im Bild)

### 7. Print

- ✓ Logo mindestens 30mm ✓ CMYK-Farben von Marketingabteilung erhalten ✓ Beschnitt 3-5mm beachten ✓ Wichtige Elemente 5mm+ vom Rand

### 8. E-Mail-Marketing

- ✓ Logo 120-150px im Header ✓ CTA-Button in Hydrophon Blau ✓ Schriftgröße mindestens 14px ✓ Mobile-optimiert

### 9. Kommunikationsstil

- ✓ Sie-Form verwenden ✓ Sachlich und professionell ✓ Technisch kompetent, aber verständlich ✗ Keine Umgangssprache ✗ Keine übertriebene Werbesprache

### 10. Markenwerte

- ✓ Innovation, Qualität, Expertise, Zukunftsorientiert ✓ Modern, aber nicht trendig ✓ Professionell, aber nicht steif
- 

## Checkliste vor Veröffentlichung

Bevor Sie Materialien veröffentlichen, prüfen Sie:

Logo:

- Richtige Variante für Hintergrund gewählt?
- Mindestgröße eingehalten (120px digital / 30mm print)?
- Schutzraum 1X frei?
- Logo nicht verzerrt, rotiert oder mit Effekten?

Farben:

- Hydrophon Blau und Grau korrekt verwendet?
- Produktlinien-Farben nur als Akzente (falls zutreffend)?
- Ausreichend Kontrast für Lesbarkeit?

- Funktionale Farben korrekt (Erfolg = Grün, Fehler = Rot)?

**Typografie:**

- Inter Schriftart verwendet?
- Schriftgrößen ausreichend (Präsentation 18px+, Social Media 20px+)?
- Klare Hierarchie (große Überschriften, kleinerer Fließtext)?

**Layout:**

- Elemente ausreichend Abstand zueinander?
- Wichtige Informationen nicht am Rand (5mm+ Abstand bei Print)?
- Mobile-Optimierung geprüft (bei digitalen Materialien)?

**Kommunikation:**

- Sie-Form verwendet?
- Sachlicher, professioneller Ton?
- Keine Rechtschreibfehler?
- Call-to-Action klar formuliert?

**Format:**

- Richtige Dateiformate (SVG/PNG/PDF)?
- Richtige Auflösung (Print: 300 DPI, Digital: 72-150 DPI)?
- CMYK-Farben für Print von Marketingabteilung erhalten?

**Bei Unsicherheit:**

- Marketingabteilung kontaktiert?
- Feedback von Kollegen eingeholt?

---

## Weitere Ressourcen

**Detaillierte Dokumentation:**

- [Logo-Guidelines](#) – Vollständige Logo-Spezifikationen mit allen Varianten
- [Farbsystem](#) – Komplette Farbpalette mit allen Nuancen und Anwendungsregeln
- [Typografie](#) – Umfassende Typografie-Spezifikationen mit allen Schriftgrößen

**Bei Fragen:**

- **Marketingabteilung:** Für Logo-Anfragen, spezielle Assets, Print-Dateien
- **Design-Team:** Für Design-Feedback, komplexe Layouts

**Feedback zu diesem Guide:** Dieser Guide wird kontinuierlich verbessert. Wenn Sie Fragen haben oder Abschnitte unklar sind, kontaktieren Sie die Marketingabteilung.

---

**Version:** 1.0 **Letzte Aktualisierung:** 2026-01-29 **Verantwortlich:** Hydrophon Marketing & Design System Team

---

# Farbsystem

---

Das Hydrophon Farbsystem bildet die visuelle Grundlage für alle Marken-Touchpoints und sorgt für eine konsistente, moderne Markenkommunikation im B2B-Sanitärbereich.

## Primärfarben

Die Primärfarben sind die Kernelemente der Hydrophon Markenidentität und wurden aus dem bestehenden Brand CI übernommen.

### Hydrophon Blau

#### Primärfarbe der Marke Hydrophon

- HEX: #008BD2
- RGB: 0, 139, 210
- **Verwendung:** Hauptfarbe für Markenelemente, Call-to-Actions, Links, primäre Buttons



### Hydrophon Grau

#### Sekundärfarbe der Marke Hydrophon

- HEX: #575656
- RGB: 87, 86, 86
- **Verwendung:** Texte, Icons, unterstützende UI-Elemente, Abgrenzungen



## Farbskala

Jede Primärfarbe verfügt über eine vollständige Skala von 50 (hellste Nuance) bis 900 (dunkelste Nuance). Der Wert 500 entspricht der Primärfarbe.

## Hydrophon Blau Skala

Stufe	HEX	Anwendung
50	#E5F5FD	Sehr helle Hintergründe, Hover-States
100	#B8E4F9	Helle Hintergründe, subtile Akzente
200	#8AD3F5	Hintergründe für Info-Boxen
300	#5CC2F1	Leichte Akzente, deaktivierte States
400	#2EB1ED	Hover-States für primäre Elemente
500	<b>#008BD2</b>	<b>Primärfarbe - Hauptanwendung</b>
600	#007BB8	Active States, Pressed States
700	#006A9E	Dunkle Akzente, Kontraste
800	#005A84	Sehr dunkle Akzente
900	#00496A	Dunkelste Nuance, maximaler Kontrast



## Hydrophon Grau Skala

Stufe	HEX	Anwendung
50	#F7F7F7	Sehr helle Hintergründe
100	#E8E8E8	Helle Hintergründe, Separatoren
200	#D1D1D1	Borders, leichte Abgrenzungen
300	#B0AFAF	Deaktivierte Elemente
400	#939292	Platzhalter-Text
500	<b>#575656</b>	<b>Primärer Text, Icons</b>
600	#4D4C4C	Dunklerer Text
700	#434242	Sehr dunkler Text
800	#393838	Nahezu schwarzer Text
900	#2F2E2E	Dunkelste Nuance



## Neutralgrau Skala

Zusätzlich zu den Markenfarben steht eine neutrale Grau-Skala für UI-Elemente zur Verfügung.

Stufe	HEX	Anwendung
50	#FAFAFA	Hellste Hintergründe
100	#F5F5F5	Sekundäre Hintergründe
200	#E5E5E5	Tertiäre Hintergründe
300	#D4D4D4	Borders
400	#A3A3A3	Platzhalter
500	#737373	Deaktivierter Text
600	#525252	Sekundärer Text
700	#404040	Primärer Text
800	#262626	Sehr dunkler Text
900	#171717	Nahezu schwarz
Weiß	FFFFFF	Reinweiß für Hintergründe
Schwarz	#000000	Reinschwarz für maximalen Kontrast

## Funktionale Farben

Funktionale Farben kommunizieren Status, Feedback und Systemzustände. Sie sind unabhängig von den Markenfarben gewählt, um Verwechslungen zu vermeiden.

### Erfolg (Success)

- HEX: #22C55E
- RGB: 34, 197, 94
- **Verwendung:** Erfolgreiche Aktionen, Bestätigungen, positive Rückmeldungen, „abgeschlossen“-Status



Erfolg

#### Anwendungsbeispiele:

- Bestätigungsnachrichten: „Formular erfolgreich gesendet“
- Status-Icons für abgeschlossene Prozesse
- Positive KPIs oder Metriken

### Warnung (Warning)

- HEX: #F59E0B
- RGB: 245, 158, 11
- **Verwendung:** Warnungen, wichtige Hinweise, die Aufmerksamkeit erfordern, „in Bearbeitung“-Status



Warnung

#### Anwendungsbeispiele:

- Hinweise auf fehlende Informationen
- Vorsichtsmaßnahmen vor Aktionen

- Status „Wartung erforderlich“

## Fehler (Error)

- HEX: #EF4444
- RGB: 239, 68, 68
- Verwendung: Fehler, Probleme, destruktive Aktionen, Validierungsfehler



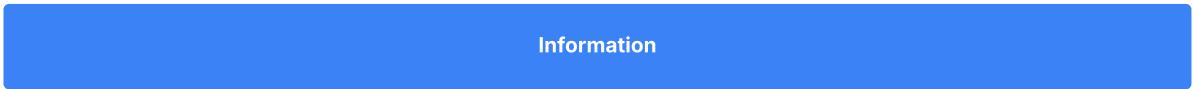
Fehler

### Anwendungsbeispiele:

- Fehlermeldungen: „E-Mail-Adresse ungültig“
- Löschen-Buttons
- Kritische Systemwarnungen

## Information (Info)

- HEX: #3B82F6
- RGB: 59, 130, 246
- Verwendung: Neutrale Informationen, Hinweise, Tooltips



Information

### Anwendungsbeispiele:

- Informationsbanner: „Neue Funktionen verfügbar“
- Hilfe-Tooltips
- Allgemeine Systemnachrichten

## Anwendungsregeln

### Primärfarbe vs. Sekundärfarbe

#### Hydrophon Blau verwenden für:

- Call-to-Action Buttons
- Links und interaktive Elemente
- Markenbezogene Hervorhebungen
- Hauptnavigation
- Primäre Icons

#### Hydrophon Grau verwenden für:

- Fließtext und Überschriften
- Icons und grafische Elemente ohne primäre Bedeutung
- Sekundäre Buttons (z.B. „Abbrechen“)
- Strukturelle UI-Elemente

## Hintergrundfarben

### Hintergrund-Hierarchie:

1. Primär: Weiß ( #FFFFFF ) - Hauptinhaltsbereiche
2. Sekundär: Neutralgrau 50 ( #FAFAFA ) - Abgrenzung von Sektionen
3. Tertiär: Neutralgrau 100 ( #F5F5F5 ) - Cards, Panels, zusätzliche Hierarchie

#### Farbige Hintergründe:

- Hydrophon Blau 50-200 für informative Bereiche mit Markenbezug
- Funktionale Farben in hellen Nuancen für Statusmeldungen

## Textfarben

### Text-Hierarchie auf weißem Hintergrund:

Verwendung	Farbe	Kontrastverhältnis
Primärer Text	Neutralgrau 900 ( #171717 )	19.6:1 (AAA)
Sekundärer Text	Neutralgrau 700 ( #404040 )	10.4:1 (AAA)
Gedämpfter Text	Neutralgrau 500 ( #737373 )	4.7:1 (AA)

### Text auf farbigen Hintergründen:

- Weißen Text ( #FFFFFF ) auf Hydrophon Blau 500: Kontrast 4.9:1 (AA) ✓
- Weißen Text auf Hydrophon Grau 500: Kontrast 7.0:1 (AAA) ✓
- Weißen Text auf allen funktionalen Farben: Mindestens AA-Standard

## Barrierefreie Farbkombinationen

Alle Farbkombinationen müssen mindestens **WCAG AA Standards** (Kontrastverhältnis  $\geq 4.5:1$  für normalen Text,  $\geq 3:1$  für großen Text) erfüllen.

### Empfohlene Kombinationen:

#### ✓ Heller Hintergrund mit dunklem Text:

- Weiß + Neutralgrau 700/800/900
- Neutralgrau 50 + Neutralgrau 700/800/900
- Hydrophon Blau 50 + Hydrophon Blau 900

#### ✓ Dunkler Hintergrund mit hellem Text:

- Hydrophon Blau 500/600/700 + Weiß
- Hydrophon Grau 500/600/700 + Weiß
- Neutralgrau 700/800/900 + Weiß

#### ✓ Funktionale Farben:

- Success/Warning/Error/Info + Weiß (auf dunklem Hintergrund)
- Helle Varianten (10-20% Opacity) + Dunkler Text

**Kontrastverhältnisse prüfen:** Nutzen Sie Tools wie [WebAIM Contrast Checker](#) zur Validierung neuer Farbkombinationen.

## Produktlinien-Farben

Die Produktlinien-Farben dienen als Referenz für produkt-spezifische Kommunikation und sollten die Marken-Primärfarben ergänzen, nicht ersetzen.

### hyHero

- Primärfarbe: Orange #F49A36 (RGB: 244, 154, 54)

- **Sekundärfarbe:** Dunkelgrau #373643 (RGB: 55, 54, 67)



hyHero Orange



hyHero Grau

**Verwendung:** Produktdatenblätter, technische Dokumentation, Produktvisualisierungen für hyHero-Linie.

## hyIndustry

- **Primärfarbe:** Dunkelblau #0E2638 (RGB: 14, 38, 56)



hyIndustry Dunkelblau

**Verwendung:** Produktdatenblätter, technische Dokumentation, Produktvisualisierungen für hyIndustry-Linie.

## Gluy

- **Primärfarbe:** Gelb #FFEEB6 (RGB: 255, 238, 182)
- **Sekundärfarbe:** Hellblau #88A9C3 (RGB: 136, 169, 195)
- **Tertiärfarbe:** Dunkelblau #14202E (RGB: 20, 32, 46)



Gluy Gelb



Gluy Hellblau



Gluy Dunkelblau

**Verwendung:** Produktdatenblätter, technische Dokumentation, Produktvisualisierungen für Gluy-Linie.

**Hinweis:** Produktlinien-Farben sollten immer in Kombination mit den Hydrophon-Primärfarben verwendet werden, um die Dachmarke zu stärken.

## Don'ts - Häufige Fehler vermeiden

### ✗ Nicht verwenden:

#### Unzureichender Kontrast:

- Hydrophon Grau 300 auf Weiß (Kontrast 2.3:1 - zu niedrig für Text)
- [Hydrophon Blau 300 auf Weiß](#) (Kontrast 2.8:1 - zu niedrig für Text)

#### Falsche funktionale Farben:

- Erfolg in Rot oder Fehler in Grün darstellen (verwirrt Nutzer)
- Hydrophon Blau für Fehlermeldungen verwenden

#### Farbkombinationen mit schlechter Lesbarkeit:

- [Hydrophon Blau auf Success Grün](#)
- [Warning Gelb auf Weiß](#) (zu niedriger Kontrast)

#### Produktlinien-Farben als Primärfarben:

- hyHero Orange nicht für primäre Buttons verwenden
- Gluy Gelb nicht als Hauphintergrund verwenden

### ✓ Stattdessen verwenden:

#### Ausreichender Kontrast:

- Neutralgrau 900 auf Weiß (19.6:1)
- Weiß auf Hydrophon Blau 500 (4.9:1)

#### Korrekte funktionale Farben:

- Erfolg immer in Grün, Fehler immer in Rot
- Info in Blau (nicht Hydrophon Blau, sondern #3B82F6)

#### Konsistente Markenkommunikation:

- Hydrophon Blau und Grau als Hauptfarben
- Produktlinien-Farben als Akzente in produktsspezifischen Kontexten

## Token-Referenz

Alle Farben sind als Design Tokens in `design-system/tokens/colors.json` im W3C DTCG Format definiert und können in Build-Systeme (Style Dictionary, Figma Tokens, etc.) importiert werden.

#### Token-Struktur:

- `hydrophon.blau.{50-900}` - Hydrophon Blau Skala
- `hydrophon.grau.{50-900}` - Hydrophon Grau Skala
- `neutral.{50-900, white, black}` - Neutrale Grau-Skala
- `color.primary` - Primärfarbe (referenziert `hydrophon.blau.500`)
- `color.secondary` - Sekundärfarbe (referenziert `hydrophon.grau.500`)
- `color.success` - Erfolgsfarbe
- `color.warning` - Warnfarbe
- `color.error` - Fehlerfarbe
- `color.info` - Infofarbe
- `color.background.{primary, secondary, tertiary}` - Hintergrundfarben
- `color.text.{primary, secondary, muted, inverse}` - Textfarben
- `productline.{hyhero, hyindustry, gluy}.*` - Produktlinien-Farben

Version: 1.0 Letzte Aktualisierung: 2026-01-28 Verantwortlich: Design System Team

# Typografie

---

Die Typografie des Hydrophon Design Systems schafft eine klare, moderne Kommunikation, die Hydrophons innovative Positionierung im B2B-Sanitärbereich widerspiegelt. Die Schrifthierarchie ist für Lesbarkeit über alle digitalen Touchpoints optimiert.

## Schriftfamilien

### Inter – Primäre Schriftfamilie

**Verwendung:** Alle Texte (Headlines, Body, UI)

Inter ist eine moderne, professionelle Sans-Serif-Schrift, die speziell für Bildschirmdarstellung optimiert wurde. Sie bietet exzellente Lesbarkeit bei allen Größen und vermittelt die gewünschte Kombination aus Innovation und technischer Glaubwürdigkeit.

**Schriftfamilie:**

```
Inter, -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, sans-serif
```

**Fallback-Stack:** Das System verwendet native Systemschriften als Fallback, um optimale Performance und Lesbarkeit auf allen Plattformen zu gewährleisten.

**Bezugsquelle:**

- Google Fonts: <https://fonts.google.com/specimen/Inter>
- Variable Font empfohlen für optimale Performance
- Benötigte Weights: 400 (Regular), 500 (Medium), 600 (Semibold), 700 (Bold)

### JetBrains Mono – Monospace-Schriftfamilie

**Verwendung:** Code-Snippets, technische Spezifikationen, Produktnummern

**Schriftfamilie:**

```
"JetBrains Mono", "Fira Code", "Courier New", monospace
```

## Typografie-Skala

Die Typografie-Skala basiert auf einer harmonischen Abstufung, die klare visuelle Hierarchien ermöglicht.

## Größenübersicht

Token	Pixel	rem	Verwendung
6xl	60px	3.75rem	Hero-Headlines, besondere Titel
5xl	48px	3rem	H1 – Hauptüberschriften
4xl	36px	2.25rem	H2 – Abschnittsüberschriften
3xl	30px	1.875rem	H3 – Unterabschnittsüberschriften
2xl	24px	1.5rem	H4 – Card-Headlines
xl	20px	1.25rem	H5 – Kleine Überschriften
lg	18px	1.125rem	H6, großer Body-Text
base	16px	1rem	Standard Body-Text
sm	14px	0.875rem	Labels, UI-Elemente
xs	12px	0.75rem	Captions, rechtliche Hinweise

## Visuelle Beispiele

6xl – 60px  
Großflächige Markenkommunikation

5xl – 48px  
H1 – Seitenhauptüberschrift

4xl – 36px  
H2 – Abschnittsüberschrift

3xl – 30px  
H3 – Unterabschnittsüberschrift

2xl – 24px  
H4 – Card- und Modulüberschrift

xl – 20px  
H5 – Kleine Überschrift

lg – 18px  
Betonter Body-Text oder H6

base – 16px  
Standard-Fließtext für alle Inhalte

**sm – 14px**  
Formular-Labels, Buttons, Navigationselemente

**xs – 12px**  
Captions, Metadaten, rechtliche Hinweise

## Hierarchie

### Überschriften (H1-H6)

Die Überschriftenhierarchie folgt semantischen HTML-Standards und bietet klare visuelle Abstufungen.

#### H1 – Hauptüberschrift (5xl)

- **Größe:** 48px (3rem)
- **Gewicht:** Bold (700)
- **Zeilenhöhe:** Tight (1.25)
- **Zeichenabstand:** Tight (-0.025em)
- **Verwendung:** Eine H1 pro Seite, definiert das Hauptthema
- **Beispiel:** "Hydrophon Produktkatalog 2026"

#### H2 – Abschnittsüberschrift (4xl)

- **Größe:** 36px (2.25rem)
- **Gewicht:** Bold (700)
- **Zeilenhöhe:** Tight (1.25)
- **Zeichenabstand:** Tight (-0.025em)
- **Verwendung:** Hauptabschnitte innerhalb einer Seite
- **Beispiel:** "Produktserien", "Technische Daten"

#### H3 – Unterabschnittsüberschrift (3xl)

- **Größe:** 30px (1.875rem)
- **Gewicht:** Semibold (600)
- **Zeilenhöhe:** Snug (1.375)
- **Zeichenabstand:** Normal (0)
- **Verwendung:** Unterabschnitte, thematische Gruppierungen
- **Beispiel:** "Sanitärmodule", "Installationshinweise"

#### H4 – Card-Überschrift (2xl)

- **Größe:** 24px (1.5rem)
- **Gewicht:** Semibold (600)
- **Zeilenhöhe:** Snug (1.375)
- **Zeichenabstand:** Normal (0)
- **Verwendung:** Überschriften in Cards, Modulen, Listen
- **Beispiel:** Produktnamen in Produktkarte

#### H5 – Kleine Überschrift (xl)

- **Größe:** 20px (1.25rem)
- **Gewicht:** Medium (500)

- **Zeilenhöhe:** Normal (1.5)
- **Zeichenabstand:** Normal (0)
- **Verwendung:** Kleinere Abschnitte, Sidebar-Headlines
- **Beispiel:** Feature-Listen, Spezifikations-Kategorien

## H6 – Kleinste Überschrift (lg)

- **Größe:** 18px (1.125rem)
- **Gewicht:** Medium (500)
- **Zeilenhöhe:** Normal (1.5)
- **Zeichenabstand:** Normal (0)
- **Verwendung:** Inline-Headlines, hervorgehobene Labels
- **Beispiel:** Formular-Abschnittsüberschriften

## Semantische HTML-Verwendung

Überschriften müssen immer semantisch korrekt verwendet werden:

- **Eine H1 pro Seite** – definiert das Hauptthema
- **Keine Hierarchie-Ebenen überspringen** – nach H2 folgt H3, nicht H4
- **Überschriften nicht für Styling verwenden** – bei Bedarf visuelle Klassen nutzen

## Body-Texte

### Body Large (lg)

- **Größe:** 18px (1.125rem)
- **Gewicht:** Regular (400)
- **Zeilenhöhe:** Relaxed (1.625)
- **Verwendung:** Einleitungstexte, hervorgehobene Absätze, Leads
- **Optimale Zeilentlänge:** 60-75 Zeichen (ca. 550-650px Breite)

### Body Base (base) – Standard

- **Größe:** 16px (1rem)
- **Gewicht:** Regular (400)
- **Zeilenhöhe:** Normal (1.5)
- **Verwendung:** Alle Standard-Fließtexte, Absätze, Beschreibungen
- **Optimale Zeilentlänge:** 60-75 Zeichen (ca. 480-560px Breite)

Dies ist die Basis-Schriftgröße für alle Body-Texte. Niemals kleiner als 16px für Hauptinhalte verwenden – dies gewährleistet Barrierefreiheit und Lesbarkeit.

### Body Small (sm)

- **Größe:** 14px (0.875rem)
- **Gewicht:** Regular (400)
- **Zeilenhöhe:** Normal (1.5)
- **Verwendung:** Sekundäre Informationen, Bildunterschriften, Metadaten
- **Optimale Zeilentlänge:** 60-75 Zeichen (ca. 420-490px Breite)

## Absatzabstände

- **Standard-Absatzabstand:** 1em (entspricht Schriftgröße)
- **Nach Überschriften:** 0.5em bis 1em
- **Vor Überschriften:** 1.5em bis 2em (außer direkt nach vorheriger Überschrift)

## Zeilenlänge und Container

Optimale Zeilenlänge für Lesbarkeit: 60-75 Zeichen

Bei größeren Viewports Textblöcke auf maximal 75 Zeichen begrenzen durch:

- Max-width auf Textcontainern
- Mehrspalten-Layout bei sehr breiten Screens
- Padding/Margin-Anpassungen

## Line Heights & Letter Spacing

### Line Heights (Zeilenhöhen)

Die Zeilenhöhe beeinflusst Lesbarkeit und visuelle Dichte maßgeblich.

Token	Wert	Verwendung
Tight	1.25	Große Headlines (H1, H2) – enge Führung für kompakte Darstellung
Snug	1.375	Subheadings (H3, H4) – ausgewogene Führung
Normal	1.5	Body-Text, kleinere Headlines – Standard-Lesbarkeit
Relaxed	1.625	Großer Body-Text – komfortable Lesbarkeit für längere Texte
Loose	2.0	Spezielle Layouts – sehr luftige Darstellung

Faustregel:

- Je größer die Schrift, desto enger die Zeilenhöhe
- Je kleiner die Schrift, desto großzügiger die Zeilenhöhe
- Längere Texte profitieren von großzügigerer Zeilenhöhe

### Letter Spacing (Zeichenabstand)

Letter Spacing (auch Tracking genannt) steuert den horizontalen Abstand zwischen Zeichen.

Token	Wert	Verwendung
Tighter	-0.05em	Extra große Display-Headlines (optional)
Tight	-0.025em	H1, H2 – Headlines wirken kompakter und kraftvoller
Normal	0em	Body-Text, H3-H6 – Standard ohne Anpassung
Wide	0.025em	Buttons, Badges – verbesserte Lesbarkeit bei UI-Elementen
Wider	0.05em	Labels in Großbuchstaben (ALL CAPS)

Wichtig: Letter Spacing bei Großbuchstaben-Text (ALL CAPS) erhöhen, mindestens +0.05em.

## UI-Elemente

### Button

- **Größe:** 14px (0.875rem)
- **Gewicht:** Medium (500)
- **Zeilenhöhe:** Normal (1.5)
- **Zeichenabstand:** Wide (0.025em)
- **Verwendung:** Alle Button-Texte
- **Hinweis:** Weiterer Zeichenabstand verbessert Lesbarkeit bei kurzen, aktionsbezogenen Texten

### Label

- **Größe:** 14px (0.875rem)
- **Gewicht:** Medium (500)
- **Zeilenhöhe:** Normal (1.5)
- **Zeichenabstand:** Normal (0)
- **Verwendung:** Formular-Labels, Navigationselemente, Tabs

### Caption

- **Größe:** 12px (0.75rem)
- **Gewicht:** Regular (400)
- **Zeilenhöhe:** Normal (1.5)
- **Zeichenabstand:** Normal (0)
- **Verwendung:** Hilfetext, Timestamps, Metadaten, rechtliche Hinweise

**Wichtig:** Caption-Text niemals kleiner als 12px verwenden – dies ist die absolute Untergrenze für Barrierefreiheit.

## Anwendungsbeispiele

### Produktseiten-Titel

```

H1 (5xl, bold, tight)
Hydrophon AquaTech Serie

H2 (4xl, bold, tight)
Technische Spezifikationen

Body Large (lg, relaxed)
Die AquaTech Serie vereint modernste Technologie mit
bewährter Qualität für höchste Ansprüche im gewerblichen
Sanitärbereich.

```

## Produkt-Card

H4 (2xl, semibold, snug)

AquaTech Pro 3000

Body Base (base, normal)

Hochleistungs-Sanitärmodul mit integrierter  
Steuerungseinheit.

Caption (xs, regular, normal)

Art.-Nr.: AT-3000-PRO

## Formular

Label (sm, medium, normal)

E-Mail-Adresse\*

Input (base, regular, normal)

[Eingabefeld]

Caption (xs, regular, normal)

Wir verwenden Ihre E-Mail nur für Auftragsbestätigungen.

## Navigationselemente

Primary Navigation

Label (sm, medium, normal)

Produkte | Lösungen | Service | Kontakt

Breadcrumb

Caption (xs, regular, normal)

Home > Produkte > AquaTech Serie

## Call-to-Action Button

Button (sm, medium, wide)

JETZT BERATEN LASSEN

## Produktspezifikationen

H3 (3xl, semibold, snug)

Abmessungen

H5 (xl, medium, normal)

Breite

Body Base (base, regular, normal)

850 mm (Standardausführung)

## Responsive Skalierung

### Desktop (1440px+)

Alle Größen wie definiert verwenden.

### Tablet (768px - 1439px)

- H1: 4xl (36px) statt 5xl (48px)
- H2: 3xl (30px) statt 4xl (36px)
- Übrige Größen unverändert

### Mobile (< 768px)

- H1: 3xl (30px) statt 5xl (48px)
- H2: 2xl (24px) statt 4xl (36px)
- H3: xl (20px) statt 3xl (30px)
- Body Text: Mindestens 16px (base) beibehalten
- Line Heights: +0.125 erhöhen für bessere Lesbarkeit auf kleinen Screens

**Wichtig:** Body-Text nie unter 16px skalieren – Barrierefreiheit hat Priorität.

## Barrierefreiheit

### WCAG-Konformität

Minimale Schriftgrößen:

- **Body-Text:** Mindestens 16px (1rem)
- **UI-Elemente:** Mindestens 14px (0.875rem)
- **Hilfstexte:** Mindestens 12px (0.75rem)

Kontrastverhältnisse:

- **Normaler Text (< 18px):** Mindestens 4.5:1 (WCAG AA)
- **Großer Text (≥ 18px):** Mindestens 3:1 (WCAG AA)
- **UI-Komponenten:** Mindestens 3:1 (WCAG AA)

## Zeilenlänge

- **Maximum:** 75 Zeichen für optimale Lesbarkeit

- Minimum: 45 Zeichen (auf mobilen Geräten akzeptabel)

## Anpassbare Schriftgrößen

Das System muss Browserskalierung unterstützen:

- Verwendung von `rem` statt `px` in CSS-Implementierung
- Keine festen `max-height` Einschränkungen auf Textcontainern
- Responsive Container, die wachsen können

## Fokus-Indikatoren

Bei interaktiven Textelementen (Links, Buttons):

- Deutlich sichtbarer Fokus-Indikator
- Mindestens 2px Outline-Breite
- Kontrast mindestens 3:1 zum Hintergrund

## Technische Implementierung

### Web Font Loading

Empfohlene Strategie:

```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700&display=swap" rel='
```

CSS Font-Face (Self-Hosted):

```
@font-face {
  font-family: 'Inter';
  src: url('/fonts/Inter-Regular.woff2') format('woff2');
  font-weight: 400;
  font-style: normal;
  font-display: swap;
}
```

**Font-Display:** `swap` verwenden für optimale Performance – Fallback-Font wird sofort angezeigt, bis Web Font geladen ist.

## CSS Custom Properties

```
:root {  
  /* Font Families */  
  --font-sans: "Inter", -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, sans-serif;  
  --font-mono: "JetBrains Mono", "Fira Code", "Courier New", monospace;  
  
  /* Font Sizes */  
  --font-size-xs: 0.75rem;      /* 12px */  
  --font-size-sm: 0.875rem;    /* 14px */  
  --font-size-base: 1rem;      /* 16px */  
  --font-size-lg: 1.125rem;    /* 18px */  
  --font-size-xl: 1.25rem;     /* 20px */  
  --font-size-2xl: 1.5rem;     /* 24px */  
  --font-size-3xl: 1.875rem;   /* 30px */  
  --font-size-4xl: 2.25rem;    /* 36px */  
  --font-size-5xl: 3rem;       /* 48px */  
  --font-size-6xl: 3.75rem;    /* 60px */  
}
```

## Token-Referenzen

Alle Werte sind in `design-system/tokens/typography.json` definiert und können mit Style Dictionary transformiert werden in:

- CSS Custom Properties
- SCSS Variables
- JavaScript/TypeScript Konstanten
- iOS/Android native Tokens

---

Version: 1.0 Letzte Aktualisierung: Januar 2026 Verantwortlich: Hydrophon Design System Team

---

# Spacing & Grid System

---

Das Spacing- und Grid-System bildet die Grundlage für konsistente räumliche Beziehungen im Hydrophon Design System.

## Inhaltsverzeichnis

- [Spacing-Skala](#)
  - [Spacing-Anwendung](#)
  - [Grid-System](#)
  - [Breakpoints](#)
  - [Gutters & Margins](#)
  - [Responsives Verhalten](#)
- 

## Spacing-Skala

Das Spacing-System basiert auf einer **4px Base Unit**. Alle Abstände sind Vielfache dieser Grundeinheit, um ein harmonisches, rhythmisches Layout zu gewährleisten.

## Komplette Skala

Token	Wert	Multiplikator	Verwendung
spacing.0	0px	0x	Kein Abstand
spacing.0.5	2px	0.5x	Extra feine Abstände
spacing.1	4px	1x	<b>Base Unit - Kleinster Abstand</b>
spacing.1.5	6px	1.5x	Minimale Abstände
spacing.2	8px	2x	Extra kleine Abstände
spacing.2.5	10px	2.5x	Kleine-mittlere Abstände
spacing.3	12px	3x	Kleine Abstände
spacing.4	16px	4x	<b>Standard Padding</b>
spacing.5	20px	5x	Mittlere Abstände
spacing.6	24px	6x	Große Abstände
spacing.7	28px	7x	Extra große Abstände
spacing.8	32px	8x	2XL Abstände
spacing.9	36px	9x	2.5XL Abstände
spacing.10	40px	10x	3XL Abstände
spacing.11	44px	11x	3.5XL Abstände
spacing.12	48px	12x	4XL Abstände
spacing.14	56px	14x	5XL Abstände
spacing.16	64px	16x	6XL Abstände - Sektionen
spacing.20	80px	20x	7XL Abstände
spacing.24	96px	24x	8XL Abstände
spacing.28	112px	28x	9XL Abstände
spacing.32	128px	32x	10XL Abstände - große Sektionen

## Visuelle Darstellung

```

spacing.1  ■ (4px)
spacing.2  ■■ (8px)
spacing.4  ■■■ (16px)
spacing.6  ■■■■ (24px)
spacing.8  ■■■■■ (32px)
spacing.12 ■■■■■■ (48px)
spacing.16 ■■■■■■■ (64px)

```

---

## Spacing-Anwendung

### Component Padding

#### Buttons:

- Kompakte Buttons: padding: spacing.2 spacing.4 (8px 16px)
- Standard Buttons: padding: spacing.3 spacing.6 (12px 24px)
- Große Buttons: padding: spacing.4 spacing.8 (16px 32px)

#### Inputs:

- Kompakte Inputs: padding: spacing.2 spacing.3 (8px 12px)
- Standard Inputs: padding: spacing.3 spacing.4 (12px 16px)
- Große Inputs: padding: spacing.4 spacing.6 (16px 24px)

#### Cards:

- Kompakte Cards: padding: spacing.4 (16px)
- Standard Cards: padding: spacing.6 (24px)
- Große Cards: padding: spacing.8 (32px)

### Section Margins

#### Zwischen Komponenten:

- Eng: spacing.2 - spacing.4 (8px - 16px)
- Standard: spacing.6 - spacing.8 (24px - 32px)
- Weit: spacing.12 - spacing.16 (48px - 64px)

#### Zwischen Sektionen:

- Standard: spacing.12 - spacing.16 (48px - 64px)
- Große Sektionen: spacing.20 - spacing.32 (80px - 128px)

### Gap zwischen Elementen

#### Listen & Gruppen:

- Kompakte Listen: gap: spacing.1 - spacing.2 (4px - 8px)
- Standard Listen: gap: spacing.3 - spacing.4 (12px - 16px)
- Luftige Listen: gap: spacing.6 - spacing.8 (24px - 32px)

#### Grid/Flex Layouts:

- Kompakt: gap: spacing.4 (16px)
- Standard: gap: spacing.6 (24px)
- Luftig: gap: spacing.8 - spacing.12 (32px - 48px)

---

## Grid-System

Das Hydrophon Grid-System basiert auf einem **12-Spalten-Layout**, das maximale Flexibilität für verschiedene Layouts bietet.

## Grundprinzipien

- **12 Spalten** ermöglichen Layouts mit 2, 3, 4, 6 oder 12 Spalten
- **Responsive Gutters** passen sich an die Bildschirmgröße an
- **Fluid Width** mit maximaler Breite von 1280px
- **Mobile-First Ansatz**

## Grid-Konfiguration

Spalten: 12

Max Width: 1280px

Gutters (Spaltenabstand):

- Mobile: 16px (spacing.4)
- Tablet: 24px (spacing.6)
- Desktop: 32px (spacing.8)

Margins (Seitenabstand):

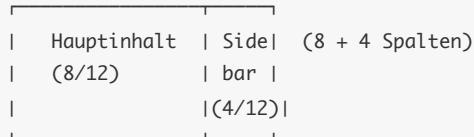
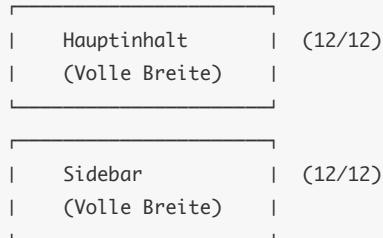
- Mobile: 16px
- Tablet: 32px
- Desktop: 64px

## Column Spans

### Häufige Spalten-Layouts:

Spalten	Breite	Verwendung
12/12	100%	Volle Breite
6/12	50%	Zwei-Spalten-Layout
4/12	33.33%	Drei-Spalten-Layout
3/12	25%	Vier-Spalten-Layout
8/12	66.66%	Hauptinhalt (2:1 Ratio)
9/12	75%	Hauptinhalt (3:1 Ratio)

### Beispiel: Artikel mit Sidebar

**Desktop:****Mobile:**

## Breakpoints

Das System verwendet einen **Mobile-First Ansatz**. Styles werden zuerst für mobile Geräte definiert und dann für größere Bildschirme erweitert.

### Breakpoint-Definitionen

Token	Wert	Gerätetyp	Beschreibung
<code>breakpoints.sm</code>	640px	Landscape Phones	Querformat Smartphones
<code>breakpoints.md</code>	768px	Tablets	Tablets im Hochformat
<code>breakpoints.lg</code>	1024px	Desktops	Kleinere Desktop-Monitore
<code>breakpoints.xl</code>	1280px	Large Desktops	Standard Desktop-Monitore
<code>breakpoints.2xl</code>	1536px	Extra Large	Große Desktop-Monitore

### Container Max-Widths

Container begrenzen die maximale Inhaltsbreite für optimale Lesbarkeit:

Breakpoint	Container Max-Width
<code>&lt; sm</code>	100% (fluid)
<code>≥ sm</code>	640px
<code>≥ md</code>	768px
<code>≥ lg</code>	1024px
<code>≥ xl</code>	1280px

### Media Query Beispiele

CSS:

```

/* Mobile first - Base styles */
.element {
  padding: 16px;
}

/* ≥ Tablet */
@media (min-width: 768px) {
  .element {
    padding: 24px;
  }
}

/* ≥ Desktop */
@media (min-width: 1024px) {
  .element {
    padding: 32px;
  }
}

```

## Gutters & Margins

### Gutter-Größen (Spaltenabstände)

Gutters sind die Abstände zwischen Grid-Spalten:

Breakpoint	Gutter	Token
Mobile (< 768px)	16px	spacing.4
Tablet (≥ 768px)	24px	spacing.6
Desktop (≥ 1024px)	32px	spacing.8

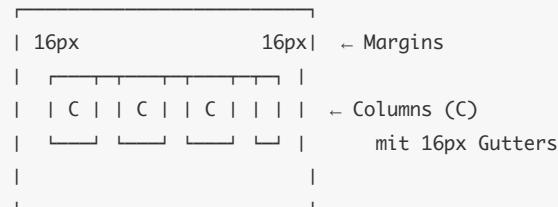
### Page Margins (Seitenabstände)

Margins sind die Abstände vom Bildschirmrand:

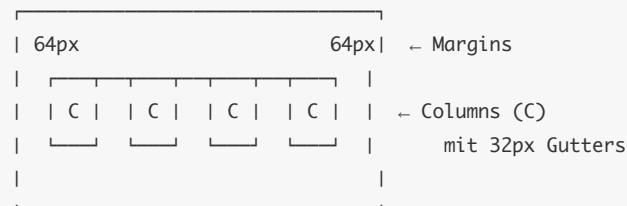
Breakpoint	Margin
Mobile (< 768px)	16px
Tablet (≥ 768px)	32px
Desktop (≥ 1024px)	64px

## Visuelle Darstellung

Mobile (< 768px):



Desktop ( $\geq 1024\text{px}$ ):

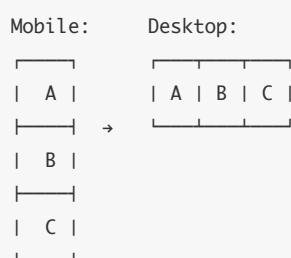


## Responsives Verhalten

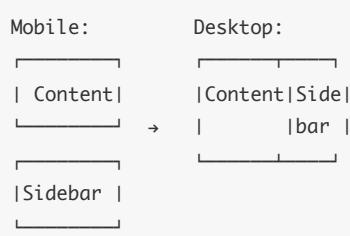
### Layout-Anpassungen über Breakpoints

Typische Responsive Patterns:

#### 1. Stack to Grid

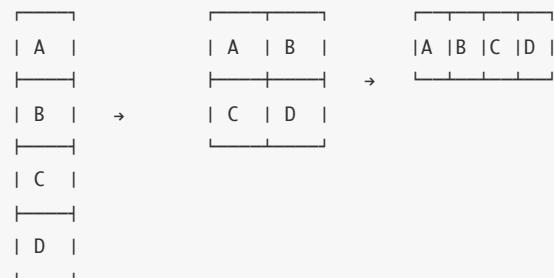


#### 2. Sidebar Collapse



#### 3. Column Reflow

Mobile (1 col): Tablet (2 cols): Desktop (4 cols):



## Spacing-Anpassungen

Padding responsive skalieren:

```

.card {
  padding: 16px;           /* spacing.4 - Mobile */
}

@media (min-width: 768px) {
  .card {
    padding: 24px;           /* spacing.6 - Tablet */
  }
}

@media (min-width: 1024px) {
  .card {
    padding: 32px;           /* spacing.8 - Desktop */
  }
}
  
```

Section Margins responsive:

```

.section {
  margin-bottom: 48px;           /* spacing.12 - Mobile */
}

@media (min-width: 1024px) {
  .section {
    margin-bottom: 64px;           /* spacing.16 - Desktop */
  }
}
  
```

## Best Practices

1. **Mobile First:** Styles zuerst für kleine Bildschirme definieren
2. **Progressive Enhancement:** Features für größere Bildschirme hinzufügen
3. **Content First:** Breakpoints an Inhalt orientieren, nicht nur an Geräten
4. **Konsistente Gutters:** Grid-Gutters sollten zu Spacing-Scale passen
5. **Lesbare Zeilenlänge:** Maximale Breite für Text-Content (z.B. 65-75 Zeichen)

## Zusammenfassung

### Spacing-System:

- 4px Base Unit für harmonische Abstände
- 22 Spacing-Werte von 0px bis 128px
- Konsistente Anwendung auf Padding, Margin und Gap

### Grid-System:

- 12-Spalten-Layout für flexible Layouts
- Responsive Gutters (16px/24px/32px)
- Max-Width 1280px für Desktop

### Breakpoints:

- 5 Breakpoints von sm (640px) bis 2xl (1536px)
- Mobile-First Ansatz
- Container Max-Widths für jedes Breakpoint

**Verwendung:** Designer und Entwickler nutzen diese Tokens für konsistente räumliche Beziehungen in allen Komponenten und Layouts.

---

# Effects: Border Radius & Schatten

---

Die Effects im Hydrophon Design System umfassen Border Radius und Schatten (Elevation), die visuelle Tiefe und Hierarchie schaffen.

## Inhaltsverzeichnis

- [Border Radius](#)
- [Schatten \(Elevation\)](#)
- [Elevation-Hierarchie](#)
- [Kombinationen](#)
- [Barrierefreiheit](#)

---

## Border Radius

Border Radius definiert die Rundung von Ecken und trägt zum modernen, zugänglichen Look des Design Systems bei.

### Border Radius Skala

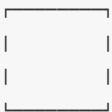
Token	Wert	Verwendung	Beispiel-Komponenten
borderRadius.none	0px	Keine Rundung	Divider, scharfe Kanten
borderRadius.sm	2px	Subtile Rundung	Badges, Tags
borderRadius.base	4px	Standard	<b>Buttons, Inputs</b>
borderRadius.md	6px	Medium	Cards, kleinere Panels
borderRadius.lg	8px	Groß	<b>Große Cards, Modals</b>
borderRadius.xl	12px	Extra groß	Pills, große Elemente
borderRadius.2xl	16px	2XL	Feature Cards
borderRadius.3xl	24px	3XL	Dekorative Elemente
borderRadius.full	9999px	Vollständig rund	<b>Avatare, Pill-Buttons</b>

## Visuelle Beispiele

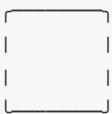
none (0px):



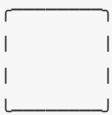
sm (2px):



base (4px):



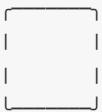
md (6px):



lg (8px):



xl (12px):



2xl (16px):



full (9999px):



## Verwendungsrichtlinien

Buttons:

- Standard-Buttons: `borderRadius.base` (4px)
- Pill-Buttons: `borderRadius.full` (9999px)
- Primäre Call-to-Action: `borderRadius.base` oder `borderRadius.md`

#### Inputs & Forms:

- Text Inputs: `borderRadius.base` (4px)
- Textareas: `borderRadius.base` (4px)
- Select Dropdowns: `borderRadius.base` (4px)

#### Cards:

- Kleine Cards: `borderRadius.md` (6px)
- Standard Cards: `borderRadius/lg` (8px)
- Feature Cards: `borderRadius.2xl` (16px)

#### Badges & Tags:

- Kompakte Badges: `borderRadius.sm` (2px)
- Pill-Badges: `borderRadius.full` (9999px)

#### Modals & Dialogs:

- Modals: `borderRadius.lg` (8px)
- Tooltips: `borderRadius.md` (6px)

#### Avatare:

- Runde Avatare: `borderRadius.full` (9999px)
- Quadratische Avatare: `borderRadius.md` (6px)

## Konsistenz-Regeln

- 1. Einheitliche Rundung innerhalb Komponente:** Verwende dieselbe Rundung für alle Ecken einer Komponente (außer bei speziellen Design-Anforderungen)
  - 2. Hierarchische Konsistenz:** Ähnliche Komponenten sollten ähnliche Border Radius verwenden
    - Alle Buttons: `borderRadius.base`
    - Alle Cards: `borderRadius.lg` oder `borderRadius.md`
  - 3. Nicht mischen:** Vermeide es, zu viele verschiedene Border Radius-Werte auf einer Seite zu verwenden
- 

## Schatten (Elevation)

Schatten erzeugen visuelle Tiefe und kommunizieren die Hierarchie und Interaktivität von Elementen. Das Hydrophon Shadow System verwendet moderne, subtile Schatten für einen professionellen B2B-Look.

## Shadow Skala

Token	Wert	Elevation	Verwendung
shadow.none	none	0	Flache Elemente
shadow.sm	0 1px 2px rgba(0,0,0,0.05)	1	Minimale Tiefe
shadow.base	0 1px 3px rgba(0,0,0,0.1), 0 1px 2px -1px rgba(0,0,0,0.1)	2	Standard Elevation
shadow.md	0 4px 6px -1px rgba(0,0,0,0.1), 0 2px 4px -2px rgba(0,0,0,0.1)	3	Erhöhte Elemente
shadow/lg	0 10px 15px -3px rgba(0,0,0,0.1), 0 4px 6px -4px rgba(0,0,0,0.1)	4	Schwebende Elemente
shadow.xl	0 20px 25px -5px rgba(0,0,0,0.1), 0 8px 10px -6px rgba(0,0,0,0.1)	5	Modals, Dialogs
shadow.2xl	0 25px 50px -12px rgba(0,0,0,0.25)	6	Dramatische Elevation
shadow.inner	inset 0 2px 4px rgba(0,0,0,0.05)	-	Eingepresste Elemente

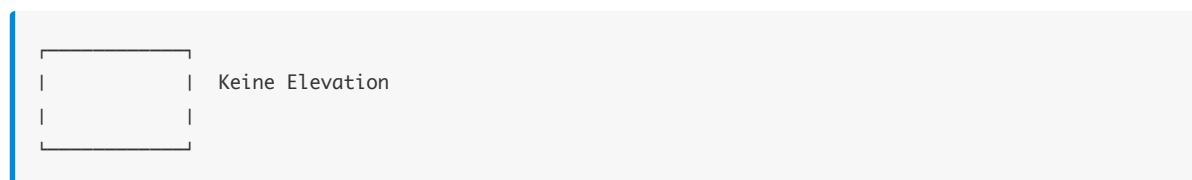
## Semantische Elevation Tokens

Für konsistente Verwendung gibt es semantische Tokens:

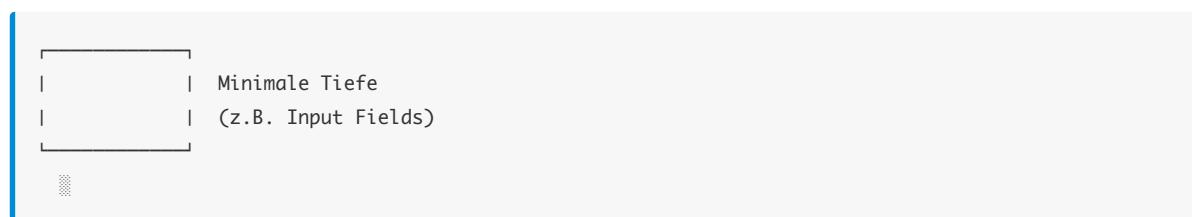
Token	Referenz	Verwendung
elevation.card	shadow.base	Cards im Ruhezustand
elevation.cardHover	shadow.md	Cards beim Hover
elevation.dropdown	shadow.lg	Dropdown-Menüs
elevation.modal	shadow.xl	Modale Dialoge
elevation.tooltip	shadow.md	Tooltips

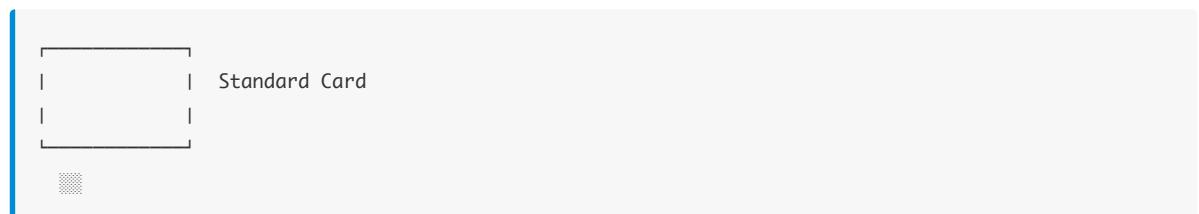
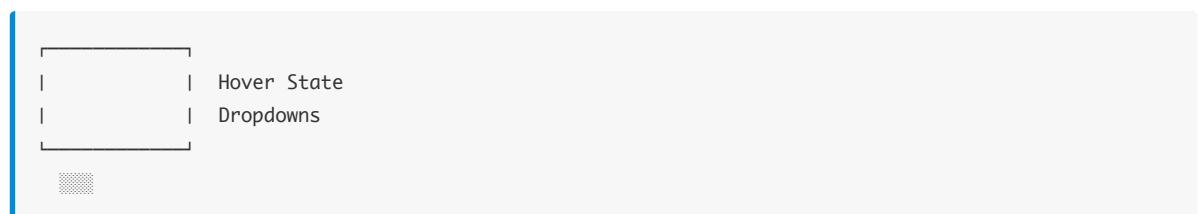
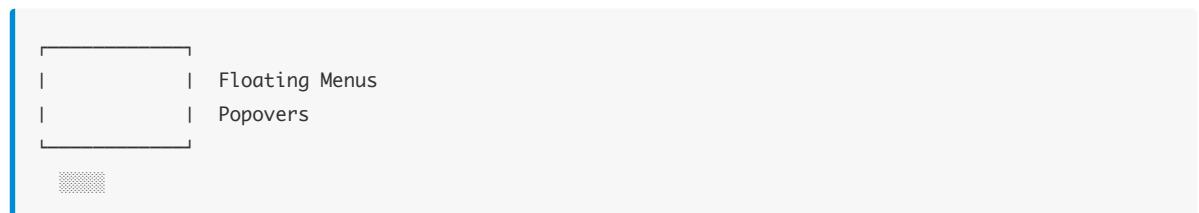
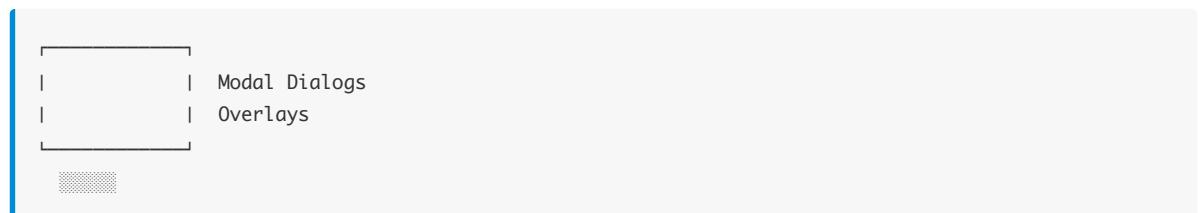
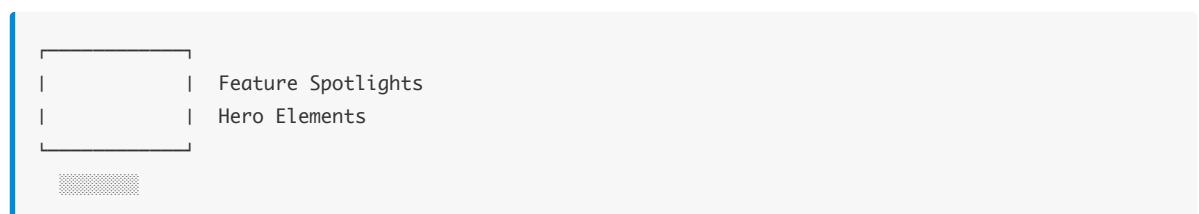
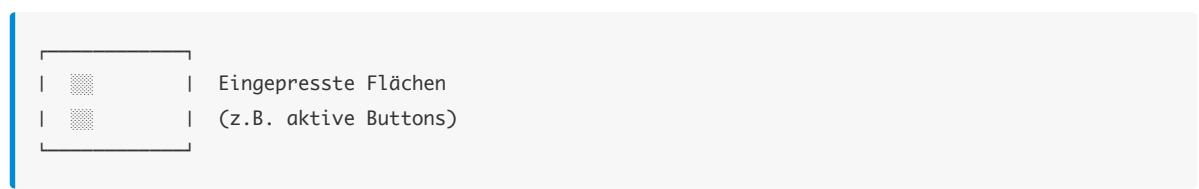
## Visuelle Beispiele

Shadow None:



Shadow SM (Subtle):



**Shadow Base (Standard):****Shadow MD (Elevated):****Shadow LG (Floating):****Shadow XL (Modal):****Shadow 2XL (Dramatic):****Shadow Inner:****Verwendungsrichtlinien****Inputs:**

- Ruhezustand: shadow.sm (subtile Tiefe)
- Fokus: shadow.md + Border (erhöhte Aufmerksamkeit)

**Cards:**

- Standard: elevation.card (shadow.base)
- Hover: elevation.cardHover (shadow.md)
- Aktiv/Ausgewählt: shadow.lg (erhöhte Elevation)

**Dropdowns & Menus:**

- Dropdown-Menüs: elevation.dropdown (shadow.lg)
- Context-Menüs: shadow.lg
- Mega-Menus: shadow.xl

**Modals & Dialogs:**

- Standard Modals: elevation.modal (shadow.xl)
- Full-Screen Overlays: shadow.2xl

**Tooltips:**

- Tooltips: elevation.tooltip (shadow.md)
- Popovers: shadow.md oder shadow.lg

**Buttons:**

- Flache Buttons: shadow.none
- Raised Buttons: shadow.sm
- Floating Action Buttons: shadow.md (hover: shadow.lg )

## Elevation-Hierarchie

Schatten kommunizieren die Ebene eines Elements im visuellen Stapel. Je höher die Elevation, desto näher erscheint das Element zum Nutzer.

### Z-Depth Konzept

Layer 6 (2xl):	Feature Spotlights, Hero
Layer 5 (xl):	Modals, Full Dialogs
Layer 4 (lg):	Floating Menus, Dropdowns
Layer 3 (md):	Hover States, Tooltips
Layer 2 (base):	Cards, Panels
Layer 1 (sm):	Inputs, subtle depth
Layer 0 (none):	Page Background, Flat Elements

### Interaktivität durch Schatten

Schatten kommunizieren Interaktivität:

**Statische Elemente:**

- Keine oder minimale Schatten ( `none` oder `sm` )
- Beispiel: Text-Container, statische Panels

**Klickbare Elemente:**

- Leichte Schatten im Ruhezustand ( `base` )
- Erhöhte Schatten beim Hover ( `md` )
- Beispiel: Cards, Buttons

**Schwebende/Floating Elemente:**

- Starke Schatten ( `lg` oder `xl` )
- Beispiel: Dropdown-Menüs, Modals

**Layering Order (Z-Index)**

Die visuelle Schichtung sollte der Elevation entsprechen:

```
Z-Index Empfehlungen:
- Base content:      z-index: 0
- Cards:             z-index: 1
- Sticky elements:  z-index: 10
- Dropdowns:          z-index: 100
- Tooltips:           z-index: 200
- Modals:             z-index: 1000
- Toast/Alerts:       z-index: 2000
```

**Kombinationen**

Empfohlene Kombinationen von Border Radius und Shadow für häufige Komponenten.

**Cards**

Standard Card:

```
border-radius: borderRadius.lg (8px)
box-shadow: elevation.card (shadow.base)
```

Card Hover:

```
border-radius: borderRadius.lg (8px)
box-shadow: elevation.cardHover (shadow.md)
transition: box-shadow 0.2s ease
```

Feature Card:

```
border-radius: borderRadius.2xl (16px)  
box-shadow: shadow.lg
```

## Buttons

Primary Button:

```
border-radius: borderRadius.base (4px)  
box-shadow: shadow.sm (optional)
```

Primary Button Hover:

```
border-radius: borderRadius.base (4px)  
box-shadow: shadow.md  
transition: box-shadow 0.15s ease
```

Pill Button:

```
border-radius: borderRadius.full (9999px)  
box-shadow: shadow.none oder shadow.sm
```

## Inputs

Text Input:

```
border-radius: borderRadius.base (4px)  
box-shadow: shadow.sm
```

Input Focus:

```
border-radius: borderRadius.base (4px)  
box-shadow: shadow.md + border-color (focus ring)
```

## Modals

Modal Dialog:

```
border-radius: borderRadius.lg (8px)  
box-shadow: elevation.modal (shadow.xl)
```

Backdrop:

```
background: rgba(0, 0, 0, 0.5)
backdrop-filter: blur(4px) (optional)
```

## Dropdowns

Dropdown Menu:

```
border-radius: borderRadius.md (6px)
box-shadow: elevation.dropdown (shadow.lg)
```

## Hover State Transitions

Empfohlene Transition:

```
transition: box-shadow 0.2s ease, transform 0.2s ease;
```

Beispiel Card Hover:

```
.card {
  box-shadow: elevation.card;
  transition: box-shadow 0.2s ease, transform 0.2s ease;
}

.card:hover {
  box-shadow: elevation.cardHover;
  transform: translateY(-2px);
}
```

## Focus States

Für Barrierefreiheit sollten Focus States zusätzlich zu Schatten einen sichtbaren Fokus-Ring haben:

```
.button:focus-visible {
  outline: 2px solid [primary-color];
  outline-offset: 2px;
  box-shadow: shadow.md;
}
```

## Barrierefreiheit

### Schatten sind nicht der einzige Indikator

Schatten allein sollten **nicht** der einzige Indikator für Interaktivität oder Hierarchie sein:

1. **Kombiniere mit Borders:** Nutze zusätzlich Border oder Outline für bessere Sichtbarkeit
2. **Farbkontrast:** Stelle sicher, dass Elemente auch ohne Schatten unterscheidbar sind
3. **Focus States:** Verwende deutliche Focus-Indikatoren (nicht nur Schatten)

## Low-Vision Unterstützung

Für Nutzer mit eingeschränktem Sehvermögen:

**Border-Alternativen:**

```
/* Zusätzlich zu Schatten */
.card {
  box-shadow: elevation.card;
  border: 1px solid rgba(0, 0, 0, 0.1);
}
```

**High Contrast Modus:**

```
@media (prefers-contrast: high) {
  .card {
    border: 2px solid currentColor;
    box-shadow: none; /* Schatten in High Contrast ausblenden */
  }
}
```

## Reduced Motion

Für Nutzer, die Animationen reduzieren möchten:

```
@media (prefers-reduced-motion: reduce) {
  .card {
    transition: none;
  }
}
```

## Focus-Visible

Nutze `:focus-visible` statt `:focus` für bessere Tastatur-Navigation:

```
.button:focus-visible {
  outline: 2px solid [primary-color];
  outline-offset: 2px;
}
```

## ARIA und Semantik

Schatten sind rein visuell. Stelle sicher, dass semantische HTML-Elemente und ARIA-Attribute die Bedeutung kommunizieren:

```
<!-- Gut: Semantisches HTML -->
<button aria-pressed="false">...</button>

<!-- Schlecht: Nur visuelles Styling -->
<div class="button-like">...</div>
```

## Zusammenfassung

### Border Radius:

- 9 Werte von `none` (0px) bis `full` (9999px)
- Standard für Buttons und Inputs: `base` (4px)
- Standard für Cards: `lg` (8px)
- Vollständig rund für Avatare: `full` (9999px)

### Schatten:

- 7 Elevation-Level plus `inner` für eingepresste Elemente
- Semantische Tokens für konsistente Verwendung
- Subtile Schatten für professionellen B2B-Look

### Best Practices:

- Kombiniere Border Radius und Schatten konsistent
- Nutze Hover-Transitions für Interaktivität
- Schatten nicht als einziger Indikator für Hierarchie
- Focus States für Barrierefreiheit

**Verwendung:** Designer und Entwickler nutzen diese Effects-Tokens für konsistente visuelle Tiefe und moderne UI-Elemente.

# Logo-Richtlinien

---

Dieses Dokument definiert die korrekte Verwendung aller Hydrophon-Logos und Produktlinien-Logos. Die Einhaltung dieser Richtlinien gewährleistet eine konsistente und professionelle Markenpräsenz über alle Touchpoints hinweg.

---

## 1. Hydrophon Hauptlogo

Das Hydrophon-Logo ist unser primäres Markenzeichen und muss korrekt und konsistent eingesetzt werden.

### Logo-Varianten

Wir bieten fünf Logo-Varianten für verschiedene Anwendungsfälle:

#### Original (Blau + Grau)



**Verwendung:** Standard-Variante für helle Hintergründe (Weiß, helles Grau). Dies ist die bevorzugte Variante für die meisten Anwendungen.

#### Einsatzbereich:

- Website-Header auf weißem Hintergrund
- Geschäftspapiere (Briefpapier, Visitenkarten)
- Produktverpackungen mit hellem Hintergrund
- Präsentationen auf weißen Folien

## Weiß

**Verwendung:** Für dunkle Hintergründe oder farbige Hintergründe, bei denen maximaler Kontrast erforderlich ist.

**Einsatzbereich:**

- Website-Bereiche mit Hydrophon Blau (#1D50A0) Hintergrund
- Dunkle Hero-Bereiche
- Videoinhalte mit dunklem Hintergrund
- Social Media Posts mit farbigem Hintergrund

Schwarz



**Verwendung:** Einfarbiger Schwarzweiß-Druck, wenn keine Farbdruckoption verfügbar ist.

**Einsatzbereich:**

- Formulare und Verträge
- Faxvorlagen
- Schwarz-Weiß-Zeitungsanzeigen
- Stempel

Blau (Einfarbig)



**Verwendung:** Wenn nur die Markenfarbe Blau verfügbar ist (einfarbiger Druck, Prägung, Gravur).

**Einsatzbereich:**

- Einfarb-Druckverfahren
- Gravuren auf Produkten
- Stickereien
- Prägedruck

## Grau (Einfarbig)



**Verwendung:** Subtile oder zurückhaltende Anwendungen, bei denen das Logo weniger dominant sein soll.

### Einsatzbereich:

- Wasserzeichen in Dokumenten
- Hintergrundmuster
- Footer-Bereiche
- Technische Dokumentation

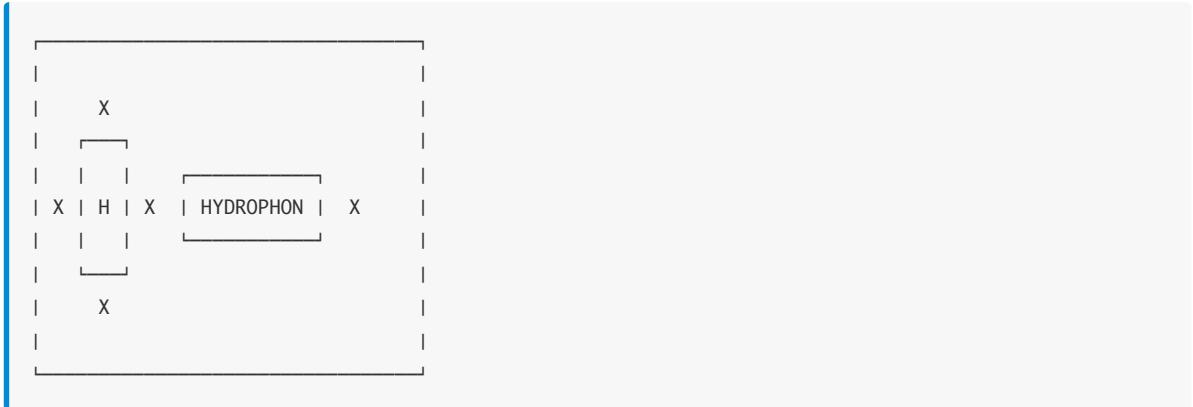
---

## Schutzraum (Clearspace)

Der Schutzraum ist der Mindestabstand, der das Logo von anderen grafischen Elementen, Texten oder Bildrändern trennen muss. Dies gewährleistet die Sichtbarkeit und Wirkung des Logos.

**Einheit:** Die Schutzraum-Einheit "X" entspricht der Höhe des Buchstabens "H" in "Hydrophon".

**Mindest-Schutzraum:** 1X auf allen Seiten (oben, unten, links, rechts)



#### Erweiterte Schutzraum-Empfehlung:

- Für hochsichtbare Platzierungen (Hauptheader, Hero-Bereiche, Plakate): 2X Schutzraum
- Für Bereiche mit visueller Komplexität (Fotos im Hintergrund): 2X Schutzraum

**Wichtig:** Der Schutzraum muss immer freigehalten werden. Keine anderen Elemente (Text, Bilder, Grafiken, Logos) dürfen in diesen Bereich eindringen.

## Mindestgrößen

Um die Lesbarkeit des Logos zu gewährleisten, müssen Mindestgrößen eingehalten werden:

### Digital (Bildschirme, Web, Apps)

- **Mindestbreite:** 120px
- **Empfohlene Mindestbreite:** 150px für optimale Lesbarkeit

#### Anwendung:

- Website-Header: Minimum 150px, empfohlen 200-250px
- Mobile Header: Minimum 120px
- Social Media Profilbilder: Minimum 180px
- E-Mail-Signaturen: Minimum 120px

### Print (Druckmaterialien)

- **Mindestbreite:** 30mm
- **Empfohlene Mindestbreite:** 40mm für optimale Lesbarkeit

#### Anwendung:

- Visitenkarten: Minimum 30mm, empfohlen 35-40mm
- Briefpapier: Minimum 40mm
- Broschüren: Minimum 30mm
- Plakate/Banner: Nach Proportionen skalieren

### Favicon (Browser-Tabs, App-Icons)

Für sehr kleine Darstellungen verwenden Sie die speziellen Favicon-Dateien:

- 16x16px: favicon-16x16.png
- 32x32px: favicon-32x32.png
- 48x48px: favicon-48x48.png
- 192x192px: android-chrome-192x192.png

- 512x512px: android-chrome-512x512.png

Diese Dateien befinden sich im Ordner: assets/logo/hydrophon/favicon/

**Wichtig:** Bei Breiten unter 120px (digital) bzw. 30mm (print) ist die Lesbarkeit stark eingeschränkt. Erwägen Sie in solchen Fällen, nur das Logo-Symbol (H-Icon) ohne Schriftzug zu verwenden, falls verfügbar.

---

## Platzierung

### Bevorzugte Positionen

#### Website/Digital:

- **Header:** Oben links, horizontal zentriert zum Container-Beginn
- **Hero-Bereiche:** Zentriert, mit großzügigem Schutzraum
- **Footer:** Links ausgerichtet oder zentriert
- **Sidebar:** Oben, linksbündig

#### Print:

- **Briefpapier:** Oben links, 20-25mm vom oberen Rand, 20-25mm vom linken Rand
- **Visitenkarten:** Vorne zentriert oder oben links
- **Broschüren:** Erste Seite oben links oder zentriert
- **Flyer:** Oben zentriert oder oben links

### Ausrichtungsregeln

- **Horizontale Ausrichtung:** Das Logo sollte mit anderen wichtigen Elementen (Navigation, Hauptinhalt) visuell ausgerichtet sein
  - **Vertikale Ausrichtung:** Bei horizontaler Anordnung mit Text sollte das Logo vertikal mittig zum Text ausgerichtet werden
  - **Abstand zu Seitenrändern:** Minimum 20px (digital) bzw. 15mm (print)
- 

## Hintergründe

### Genehmigte Hintergründe

Das Logo funktioniert am besten auf folgenden Hintergründen:

#### Weiß und helle Farben:

- Weiß (#FFFFFF)
- Hellgrau (#F5F5F5, #EEEEEE)
- Sehr helle Farbtöne (Pastellfarben mit hoher Helligkeit)
- **Logo-Variante:** Original (Blau + Grau)

#### Hydrophon Blau:

- Hydrophon Blau (#1D50A0)
- Dunkle Blautöne
- **Logo-Variante:** Weiß

#### Dunkle Farben:

- Dunkelgrau (#333333, #222222)
- Schwarz (#000000)
- Dunkle Farbtöne

- Logo-Variante: Weiß

### Kontrast-Anforderungen

**Minimaler Kontrast:** 4.5:1 (WCAG AA Standard) **Empfohlener Kontrast:** 7:1 (WCAG AAA Standard)

**Testen Sie den Kontrast:** Verwenden Sie Tools wie den WebAIM Contrast Checker, um sicherzustellen, dass das Logo ausreichend Kontrast zum Hintergrund hat.

### Foto- und Bildhintergründe

Wenn das Logo auf Fotos oder komplexen Bildhintergründen platziert wird:

#### Anforderungen:

1. **Vermeiden Sie visuell unruhige Bereiche** - Platzieren Sie das Logo auf ruhigen, einfarbigen Bereichen des Bildes
2. **Nutzen Sie Overlay-Techniken** - Fügen Sie einen halbtransparenten weißen oder dunklen Overlay hinzu, um den Kontrast zu verbessern
3. **Erweitern Sie den Schutzraum** - Verwenden Sie mindestens 2X Schutzraum bei Foto-Hintergründen
4. **Wählen Sie die richtige Variante** - Weiße Variante für dunkle Fotos, Original für helle Fotos

**Empfehlung:** Vermeiden Sie nach Möglichkeit die Platzierung des Logos direkt auf Fotos. Nutzen Sie stattdessen einfarbige Bereiche oder Overlay-Boxen.

---

### Fehlverwendung (Don'ts)

Die folgenden Verwendungen sind nicht gestattet und beschädigen die Markenintegrität:

#### 1. Nicht stauchen oder strecken

**✗ Falsch:** Das Logo darf nicht horizontal oder vertikal verzerrt werden.

- Dies verändert die Proportionen und macht das Logo unprofessionell
- Verwenden Sie immer proportionales Skalieren (Seitenverhältnis beibehalten)

#### 2. Nicht rotieren

**✗ Falsch:** Das Logo darf nicht gedreht oder schräg gestellt werden.

- Das Logo muss immer horizontal ausgerichtet sein
- Keine Rotation um irgendeine Achse

#### 3. Keine Schlagschatten oder Effekte

**✗ Falsch:** Keine Schlagschatten, Leuchteffekte, 3D-Effekte oder ähnliche grafische Effekte.

- Das Logo ist bereits für optimale Wirkung gestaltet
- Effekte reduzieren die Professionalität und Klarheit

#### 4. Nicht auf kontrastarmen Hintergründen platzieren

**✗ Falsch:** Das Logo darf nicht auf Hintergründen mit unzureichendem Kontrast platziert werden.

- Blaues Logo nicht auf blauem Hintergrund
- Graues Logo nicht auf grauem Hintergrund
- Immer mindestens 4.5:1 Kontrast sicherstellen

#### 5. Nicht beschneiden

**✗ Falsch:** Keine Teile des Logos dürfen abgeschnitten oder beschnitten werden.

- Das vollständige Logo muss immer sichtbar sein
- Auch der Schutzraum darf nicht beschnitten werden

## 6. Farben nicht ändern

**✗ Falsch:** Die Logo-Farben dürfen nicht geändert werden.

- Verwenden Sie nur die bereitgestellten Varianten
- Keine individuellen Farbkombinationen erstellen
- Keine Transparenz-Reduktion (außer für Wasserzeichen mit spezifischer Genehmigung)

## 7. Keine Effekte hinzufügen

**✗ Falsch:** Keine Verläufe, Konturen, Muster oder Texturen auf das Logo anwenden.

- Das Logo ist bereits final gestaltet
- Zusätzliche Effekte verfälschen die Markenidentität

## 8. Nicht zu nah an anderen Elementen

**✗ Falsch:** Das Logo darf nicht zu nah an anderen grafischen Elementen, Texten oder Logos platziert werden.

- Halten Sie immer den Mindest-Schutzraum ein (1X)
- Bei komplexen Layouts verwenden Sie erweiterten Schutzraum (2X)

### Weitere Verbote:

- Kein Nachzeichnen oder Nachbauen des Logos
- Keine Kombination mit anderen Logos ohne Genehmigung (siehe Co-Branding-Regeln)
- Keine Verwendung alter oder veralteter Logo-Versionen
- Keine individuellen Interpretationen oder Variationen

**Bei Unsicherheit:** Wenden Sie sich an die Marketingabteilung für Klärung.

---

## 2. Produktlinien-Logos

Hydrophon umfasst mehrere Produktlinien, jede mit eigenem Logo, das konsistent mit der Hauptmarke verwendet werden muss.

### Gluy



#### Verfügbare Varianten:

- Original: Gelbe und blaue Elemente auf transparentem Hintergrund

- **Weiß:** Für dunkle Hintergründe
- **Schwarz:** Für Schwarz-Weiß-Druck
- **Hellblau:** Einfarbige Anwendung in Hellblau
- **Dunkelblau:** Einfarbige Anwendung in Dunkelblau

**Primäre Farbkontexte:**

- Gelb: #FFEEB6 (Gluy Primärfarbe)
- Hellblau: #88A9C3 (Gluy Sekundärfarbe)

**Anwendungsbereiche:**

- Gluy-Produktverpackungen
- Gluy-Produktseiten auf der Website
- Gluy-spezifische Marketingmaterialien
- Social Media Posts zu Gluy-Produkten

**Schutzraum und Mindestgrößen:**

- Es gelten die gleichen Regeln wie für das Hydrophon-Hauptlogo
- Mindest-Schutzraum: 1X (basierend auf der Höhe des "G")
- Mindestbreite digital: 120px
- Mindestbreite print: 30mm

**Dateipfade:**

- SVG: assets/logo/gluy/svg/logo-gluy-[variante].svg
- PNG 2x: assets/logo/gluy/png/logo-gluy-[variante]@2x.png
- PNG 3x: assets/logo/gluy/png/logo-gluy-[variante]@3x.png

## hyHero



### Verfügbare Varianten:

- **Original:** Orange und Grau auf transparentem Hintergrund
- **Weiß:** Für dunkle Hintergründe
- **Schwarz:** Für Schwarz-Weiß-Druck

### Primäre Farbkontexte:

- Orange: #F49A36 (hyHero Primärfarbe)

### Anwendungsbereiche:

- hyHero-Produktverpackungen
- hyHero-Produktseiten auf der Website
- hyHero-spezifische Marketingmaterialien

- Social Media Posts zu hyHero-Produkten
- Technische Dokumentation für hyHero-Produkte

**Schutzraum und Mindestgrößen:**

- Es gelten die gleichen Regeln wie für das Hydrophon-Hauptlogo
- Mindest-Schutzraum: 1X (basierend auf der Höhe des "h")
- Mindestbreite digital: 120px
- Mindestbreite print: 30mm

**Dateipfade:**

- SVG: assets/logo/hyhero/svg/logo-hyhero-[variante].svg
  - PNG 2x: assets/logo/hyhero/png/logo-hyhero-[variante]@2x.png
  - PNG 3x: assets/logo/hyhero/png/logo-hyhero-[variante]@3x.png
- 

**hyIndustry**



**Verfügbare Varianten:**

- **Original:** Dunkelblau und Grau auf transparentem Hintergrund
- **Weiß:** Für dunkle Hintergründe
- **Schwarz:** Für Schwarz-Weiß-Druck

**Primäre Farbkontexte:**

- Dunkelblau: #0E2638 (hyIndustry Primärfarbe)

**Anwendungsbereiche:**

- hyIndustry-Produktverpackungen
- hyIndustry-Produktseiten auf der Website
- hyIndustry-spezifische Marketingmaterialien
- Social Media Posts zu hyIndustry-Produkten
- B2B-Industriedokumentation

**Schutzraum und Mindestgrößen:**

- Es gelten die gleichen Regeln wie für das Hydrophon-Hauptlogo
- Mindest-Schutzraum: 1X (basierend auf der Höhe des "h")
- Mindestbreite digital: 120px
- Mindestbreite print: 30mm

**Dateipfade:**

- SVG: assets/logo/hyindustry/svg/logo-hyindustry-[variante].svg
- PNG 2x: assets/logo/hyindustry/png/logo-hyindustry-[variante]@2x.png

- PNG 3x: assets/logo/hyindustry/png/logo-hyindustry-[variante]@3x.png

## Co-Branding: Hydrophon + Produktlinien-Logo

In bestimmten Situationen müssen das Hydrophon-Hauptlogo und ein Produktlinien-Logo gemeinsam verwendet werden.

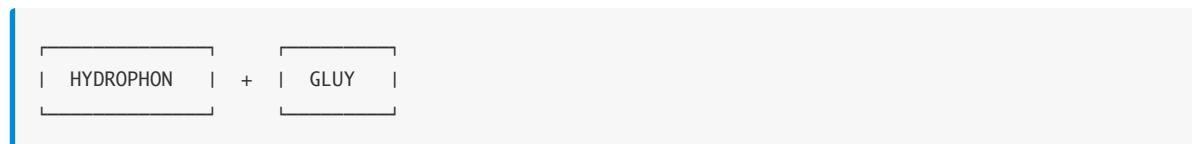
### Hierarchie-Regeln

**Hydrophon ist die Dachmarke:**

- Das Hydrophon-Logo ist immer primär
- Produktlinien-Logos sind sekundär und unterstützend

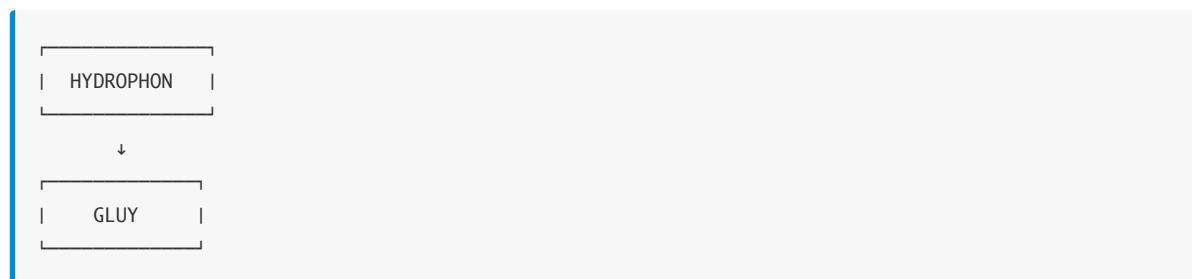
### Anordnung

**Bevorzugte Anordnung: Horizontal**



- Hydrophon-Logo links, Produktlinien-Logo rechts
- Trennzeichen in der Mitte (optional): Linie oder kein Trennzeichen
- Vertikale Zentrierung beider Logos

**Alternative Anordnung: Vertikal**



- Hydrophon-Logo oben, Produktlinien-Logo unten
- Horizontal zentriert
- Abstand: Mindestens 0.5X

### Größenverhältnis

**Hydrophon-Logo:** 100% Größe (Referenz) **Produktlinien-Logo:** 60-80% der Breite des Hydrophon-Logos

**Beispiel:**

- Hydrophon-Logo: 200px breit
- Gluy-Logo: 120-160px breit

### Abstand zwischen Logos

**Mindestabstand:** 1X (basierend auf der Schutzraum-Einheit des Hydrophon-Logos) **Empfohlener Abstand:** 1.5X

### Anwendungsbeispiele

**Wann Co-Branding verwenden:**

- Produktlinien-spezifische Landingpages (Header)

- Messe-Displays für spezifische Produktlinien
- Produktverpackungen, die zur gesamten Hydrophon-Familie gehören
- Broschüren, die mehrere Produktlinien abdecken

**Wann nur Produktlinien-Logo verwenden:**

- Produktspezifische Detailseiten (ohne Hydrophon-Hauptlogo)
- Social Media Posts, die nur ein Produkt bewerben
- Produktetiketten mit begrenztem Platz

**Wann nur Hydrophon-Logo verwenden:**

- Unternehmensweite Kommunikation
- Hauptwebsite-Header
- Allgemeine Geschäftspapiere
- Imagekampagnen

**Fehlverwendung bei Co-Branding**

**✗ Falsch:** Produktlinien-Logo größer als Hydrophon-Logo **✗ Falsch:** Zu geringer Abstand zwischen den Logos **✗ Falsch:** Logos auf unterschiedlichen Baselines (nicht vertikal ausgerichtet) **✗ Falsch:** Verwendung unterschiedlicher Varianten (z.B. Original + Schwarz)

---

### 3. Dateiformate

#### SVG (Scalable Vector Graphics)

**Verwendung:** Web, digitale Anwendungen, skalierbare Grafiken

**Vorteile:**

- Verlustfreie Skalierung auf jede Größe
- Kleine Dateigröße
- Perfekte Schärfe auf allen Bildschirmen (inkl. Retina)
- Editierbar in Vektorgrafikprogrammen

**Wann verwenden:**

- Website-Header und -Footer
- Web-Applikationen
- Responsive Designs
- Icons und UI-Elemente
- Moderne E-Mail-Clients (mit Fallback)

**Pfad:** assets/logo/[produktlinie]/svg/logo-[produktlinie]-[variante].svg

**Hinweis:** SVG ist das bevorzugte Format für alle digitalen Anwendungen, wenn möglich.

---

#### PNG (Portable Network Graphics)

**Verwendung:** Raster-Web-Grafiken, Retina-Displays, E-Mail, Social Media

**Vorteile:**

- Transparenter Hintergrund
- Breite Kompatibilität
- Optimiert für verschiedene Bildschirmauflösungen

#### Verfügbare Auflösungen:

- @2x: Für Standard-Retina-Displays (2x Pixeldichte)
- @3x: Für High-Resolution-Displays (3x Pixeldichte, z.B. iPhone)

#### Wann verwenden:

- E-Mail-Signaturen
- PowerPoint/Keynote-Präsentationen (wenn SVG nicht unterstützt wird)
- Social Media Posts und Profilbilder
- Legacy-Systeme ohne SVG-Support
- Raster-Bildbearbeitungsprogramme (Photoshop, etc.)

**Pfad:** assets/logo/[produktlinie]/png/logo-[produktlinie]-[variante]@[2x|3x].png

**Empfehlung:** Verwenden Sie @2x als Standard für moderne Webanwendungen. @3x für High-End-Mobile-Anwendungen.

---

## PDF (Portable Document Format)

**Verwendung:** Druckproduktion, professioneller Druck

**Hinweis:** PDF-Dateien für Druckzwecke sind in einem separaten Assets-Ordner verfügbar und unterscheiden sich je nach Farbraum:

- **PDF-RGB:** Für Digitaldruck und Screen-Anwendungen
- **PDF-CMYK:** Für professionellen Offset-Druck

#### Wann verwenden:

- Druckereien (Visitenkarten, Broschüren, Flyer)
- Hochwertige Printproduktionen
- Verpackungsdesign
- Großformatdruck (Banner, Plakate)

**Pfad:** Diese Dateien befinden sich im Original-Assets-Verzeichnis außerhalb des Design-Systems und werden bei Bedarf von der Marketingabteilung bereitgestellt.

---

## Favicon-spezifische Formate

Für Browser-Favicons und App-Icons werden spezielle vorbereitete PNG-Dateien verwendet:

#### Verfügbare Größen:

- **16x16:** Klassisches Browser-Favicon
- **32x32:** Standard-Favicon für moderne Browser
- **48x48:** Chrome-Adressleiste
- **192x192:** Android Chrome App-Icon
- **512x512:** Progressive Web App (PWA) Icon

#### Zusätzliche Dateien:

- `apple-touch-icon.png` - Optimierte für iOS Home-Screen
- `site.webmanifest` - Web App Manifest für PWA
- `favicon-einbindung.html` - HTML-Snippet für korrekte Einbindung

**Pfad:** assets/logo/hydrophon/favicon/

#### Einbindung im HTML:

```
<link rel="icon" type="image/png" sizes="32x32" href="/assets/logo/hydrophon/favicon/favicon-32x32.png">
<link rel="icon" type="image/png" sizes="16x16" href="/assets/logo/hydrophon/favicon/favicon-16x16.png">
<link rel="apple-touch-icon" sizes="180x180" href="/assets/logo/hydrophon/favicon/apple-touch-icon.png">
<link rel="manifest" href="/assets/logo/hydrophon/favicon/site.webmanifest">
```

## 4. Download & Verwendung

### Dateinamenskonvention

Alle Logo-Dateien folgen dieser Namensstruktur:

logo-[produktlinie]-[variante].[dateierweiterung]

#### Beispiele:

- logo-hydrophon-original.svg
- logo-gluy-weiss@2x.png
- logo-hyhero-schwarz.svg
- logo-hyindustry-original@3x.png

#### Produktlinien:

- hydrophon - Hauptmarke
- gluy - Gluy-Produktlinie
- hyhero - hyHero-Produktlinie
- hyindustry - hyIndustry-Produktlinie

#### Varianten:

- original - Mehrfarbige Standardversion
- weiss - Weiße Einfarbversion
- schwarz - Schwarze Einfarbversion
- blau - Blaue Einfarbversion (nur Hydrophon)
- grau - Graue Einfarbversion (nur Hydrophon)
- hellblau - Hellblaue Einfarbversion (nur Gluy)
- dunkelblau - Dunkelblaue Einfarbversion (nur Gluy)

### Asset-Ordner-Struktur

Die Logos sind im Repository wie folgt organisiert:

```

design-system/
└── assets/
    └── logo/
        ├── hydrophon/
        │   ├── svg/
        │   │   ├── logo-hydrophon-original.svg
        │   │   ├── logo-hydrophon-weiss.svg
        │   │   ├── logo-hydrophon-schwarz.svg
        │   │   ├── logo-hydrophon-blau.svg
        │   │   └── logo-hydrophon-grau.svg
        │   └── png/
        │       ├── logo-hydrophon-original@2x.png
        │       ├── logo-hydrophon-original@3x.png
        │       └── ... (alle Varianten)
        └── favicon/
            ├── favicon-16x16.png
            ├── favicon-32x32.png
            └── ... (alle Favicon-Dateien)
    └── gluy/
        ├── svg/
        └── png/
    └── hyhero/
        ├── svg/
        └── png/
    └── hyindustry/
        ├── svg/
        └── png/

```

## Verwendung in verschiedenen Kontexten

### Webentwicklung

- Verwenden Sie **SVG** für beste Qualität und Performance
- Nutzen Sie **PNG @2x** als Fallback für ältere Browser
- Implementieren Sie `<picture>`-Element für optimale Browser-Unterstützung

### Beispiel:

```

<picture>
  <source srcset="/assets/logo/hydrophon/svg/logo-hydrophon-original.svg" type="image/svg+xml">
    
  </picture>

```

### Design-Tools (Figma, Sketch, Adobe XD)

- Importieren Sie **SVG**-Dateien direkt
- Behalten Sie Vektorformat bei für flexible Skalierung

### Präsentationen (PowerPoint, Keynote, Google Slides)

- Verwenden Sie **PNG @2x** für beste Darstellung

- Für Druckpräsentationen: Exportieren Sie zu PDF und verwenden Sie SVG

#### E-Mail-Marketing

- Verwenden Sie **PNG @2x** (beste Kompatibilität)
- Optimieren Sie Dateigröße für schnelles Laden
- Fügen Sie immer Alt-Text hinzu

#### Social Media

- **Facebook/LinkedIn:** PNG @2x, mindestens 180×180px
- **Instagram:** PNG @3x, quadratisches Format für Profilbild
- **Twitter:** PNG @2x, 400×400px empfohlen

#### Druck

- Kontaktieren Sie die Marketingabteilung für **PDF-CMYK-Dateien**
  - Diese sind optimiert für professionellen Offsetdruck
- 

### Kontakt für Fragen und individuelle Anfragen

#### Marketingabteilung:

- Für Logo-Dateien in speziellen Formaten (z.B. EPS, AI)
- Für Anpassungen oder Sonderfälle
- Für Co-Branding mit externen Partnern
- Für Print-PDF-Dateien (RGB/CMYK)
- Bei Unsicherheiten zur korrekten Verwendung

#### Design-Team:

- Für technische Fragen zur Integration
  - Für Design-Feedback und Best Practices
  - Für Entwicklung neuer Anwendungsfälle
- 

### Zusammenfassung der wichtigsten Regeln

1. Verwenden Sie immer die bereitgestellten offiziellen Logo-Dateien - Bauen Sie das Logo nicht nach
2. Halten Sie den Mindest-Schutzraum von 1X ein - Das Logo braucht Raum zum Atmen
3. Respektieren Sie die Mindestgrößen - 120px (digital) / 30mm (print)
4. Wählen Sie die richtige Variante für den Hintergrund - Original für hell, Weiß für dunkel
5. Stauchen, strecken oder rotieren Sie das Logo nicht - Proportionen sind heilig
6. Fügen Sie keine Effekte hinzu - Das Logo ist bereits perfekt gestaltet
7. Stellen Sie ausreichend Kontrast sicher - Mindestens 4.5:1
8. Bei Co-Branding: Hydrophon ist immer primär - Produktlinien-Logos sind sekundär

Bei Fragen oder Unsicherheiten wenden Sie sich immer an die Marketingabteilung.

---

Version: 1.0 Letzte Aktualisierung: 2026-01-28 Verantwortlich: Hydrophon Marketing

---

# Icons

---

Icon-System für das Hydrophon Design System. Definiert visuellen Stil, Icon-Groessen, Bibliothek und Anwendungsrichtlinien für konsistente Icon-Nutzung über alle Touchpoints hinweg.

## Inhaltsverzeichnis

- [Icon-Stil](#)
  - [Icon-Bibliothek](#)
  - [Groessen und Anwendung](#)
  - [Barrierefreiheit](#)
  - [Export-Formate](#)
  - [Don'ts](#)
- 

## Icon-Stil

### Visuelle Spezifikationen

**Strichstaerke:** 2px

- Ausgewogen und vielseitig
- Lesbar bei allen Groessen (16px bis 48px)
- Industrie-Standard für moderne B2B-Anwendungen

**Strichabschluss:** Abgerundet (round)

- Border-radius: 2px an allen Endpunkten
- Weiche, moderne B2B-Aesthetik
- Konsistent mit den Rundungen der Inter-Schriftfamilie

**Strichverbindungen:** Abgerundet (round)

- Keine scharfen Ecken an Verbindungspunkten
- Harmonische Gesamtoptik

### Grid-System

**Basis-Grid:** 24×24px

- Standard-Artboard für alle Icons
- Industrie-Standard (Material Design, Phosphor, Lucide)
- Optimale Balance zwischen Details und Performance

**Grid-Padding:** 2px

- 1px visueller Abstand auf jeder Seite
- Sichere Zone: 20×20px für Icon-Pfade
- Verhindert Beschneidung bei verschiedenen Containergroessen

## Visuelle Darstellung



### Warum diese Wahl?

- **Moderne B2B-Aesthetik:** Abgerundete Formen wirken zugaenglich und professionell, nicht technisch-kalt
- **Konsistenz:** 2px Stroke matcht die optische Staerke der Inter-Schriftfamilie
- **Skalierbarkeit:** Von 16px (inline) bis 48px (dekorativ) gut lesbar
- **Wartbarkeit:** Industrie-Standard erleichtert Erweiterung und Custom-Icons

## Icon-Bibliothek

### Primaere Bibliothek: Lucide Icons

Lucide Icons als Basis-Bibliothek gewaeHLT:

- **Lizenz:** ISC License (permissive, kommerzielle Nutzung erlaubt)
- **Stil:** 2px Stroke, anpassbar, konsistent mit Hydrophon-Anforderungen
- **Umfang:** 1000+ Icons für alle UI-Anwendungsfaelle
- **Technische Integration:** React, Vue, SVG-Exports, Framework-agnostisch
- **Website:** <https://lucide.dev>

### Installation

React-Integration:

```
npm install lucide-react
```

**Framework-agnostische SVG-Nutzung:** Optimierte SVG-Dateien sind bereits im Design-System exportiert unter:  
design-system/assets/icons/

### Basis-Icon-Set (35 Icons)

Alle Icons sind als optimierte SVG-Dateien verfuegbar unter `design-system/assets/icons/`.

### Navigation (10 Icons)

Icons fuer Navigationselemente und Bewegung durch Interfaces.

Pfad: assets/icons/navigation/

Icon-Name	Datei	Anwendung
Menu	menu.svg	Hamburger-Menu öffnen, Hauptnavigation
Schliessen	x.svg	Dialoge schliessen, Menu zuklappen
Chevron Down	chevron-down.svg	Dropdown öffnen, Accordion ausklappen
Chevron Right	chevron-right.svg	Nächster Schritt, Navigation vorwärts
Chevron Left	chevron-left.svg	Zurück-Navigation, vorheriger Schritt
Arrow Left	arrow-left.svg	Zurück-Button, vorherige Seite
Arrow Right	arrow-right.svg	Weiter-Button, nächste Seite
Arrow Up	arrow-up.svg	Nach oben scrollen, Sort aufsteigend
Arrow Down	arrow-down.svg	Nach unten scrollen, Sort absteigend
Home	home.svg	Startseite, Dashboard

### Actions (12 Icons)

Icons für Benutzeraktionen und Operationen.

Pfad: assets/icons/actions/

Icon-Name	Datei	Anwendung
Suche	search.svg	Suchfelder, Filterung
Filter	filter.svg	Filter-Optionen, Datenansichten
Download	download.svg	Dateien herunterladen, Export
Upload	upload.svg	Dateien hochladen, Import
Teilen	share.svg	Content teilen, Link kopieren
Bearbeiten	edit.svg	Einträge bearbeiten, Formular-Edit
Loeschen	trash-2.svg	Einträge löschen, Entfernen
Hinzufügen	plus.svg	Neuer Eintrag, Element hinzufügen
Entfernen	minus.svg	Element entfernen, Reduzieren
Kopieren	copy.svg	In Zwischenablage kopieren
Bestätigen	check.svg	Aktion bestätigen, Auswahl
Externer Link	external-link.svg	Link öffnet in neuem Tab

### Status & Feedback (8 Icons)

Icons für Systemstatus, Feedback und Benachrichtigungen.

Pfad: assets/icons/status/

Icon-Name	Datei	Anwendung
Erfolg	check-circle.svg	Erfolgreiche Aktion, Bestätigung
Fehler	alert-circle.svg	Fehlermeldung, kritischer Status
Information	info.svg	Hinweis, zusätzliche Information
Warnung	alert-triangle.svg	Warnung, Aufmerksamkeit notig
Laden	loader.svg	Ladevorgang, asynchrone Operation
Abgebrochen	x-circle.svg	Aktion abgebrochen, nicht erfolgreich
Benachrichtigung	bell.svg	Neue Nachrichten, Alerts
Zeit	clock.svg	Zeitstempel, Verlauf, Pending

### Product-Specific (5 Icons)

Produkt-spezifische Icons als Platzhalter für Hydrophon-Anwendungen.

Pfad: assets/icons/product/

Icon-Name	Datei	Anwendung
Wasser	droplet.svg	Wasserbezogene Produkte, Ablauf-Systeme
Einstellungen	settings.svg	Konfiguration, Technische Parameter
Werkzeug	wrench.svg	Installation, Handwerker-Kontext
Produkt	package.svg	Produkt-Details, Verpackung
Ebenen	layers.svg	Produktkomponenten, Systemaufbau

**Hinweis für Custom Icons:** Wenn produktspezifische Icons erstellt werden (z.B. spezifische Ablaufsysteme, Zertifizierungen), sollten sie dem Lucide-Stil folgen:

- 2px Stroke
- Round Caps & Joins
- 24×24px Grid mit 2px Padding
- viewBox="0 0 24 24"
- stroke="currentColor"

### Erweiterung der Bibliothek

Neue Icons aus Lucide hinzufügen:

1. Icon-Namen auf [lucide.dev](https://lucide.dev) finden
2. SVG aus `node_modules/lucide-static/icons/{name}.svg` kopieren
3. In passende Kategorie unter `assets/icons/` ablegen
4. SVGO-Optimierung ausführen:

```
npx svgo -f assets/icons --config svgo.config.js -r
```

5. Icon in dieser Dokumentation ergänzen

## Groessen und Anwendung

### Icon-Groessen-Tabelle

Groesse	Pixel	Token	Anwendung	Beispiel
XS	16px	icon.size.xs	Inline mit kleinem Text, kompakte UIs	Breadcrumb-Chevrons, Tag-Icons
SM	20px	icon.size.sm	Buttons mit Text, Formulare	"Speichern"-Button, Input-Prefix
MD	24px	icon.size.md	Standalone-Icons, Icon-Buttons	Toolbar-Aktionen, Navigationselemente
LG	32px	icon.size.lg	Hervorgehobene Aktionen	Feature-Highlights, CTA-Buttons
XL	48px	icon.size.xl	Dekorative Elemente	Leere-State-Illustrationen, Hero-Sections

### Anwendungsrichtlinien

#### Kontextuelle Groesse:

- **Text-Inline:** Icon-Groesse sollte Line-Height des Textes matchen
  - Text 14px (line-height 20px) → Icon 16px (xs)
  - Text 16px (line-height 24px) → Icon 20px (sm)
- **Buttons:** Icon sollte visuell balanciert sein mit Text-Groesse
  - Kleine Buttons (32px Hoehe) → 16px Icon
  - Mittlere Buttons (40px Hoehe) → 20px Icon
  - Grosse Buttons (48px Hoehe) → 24px Icon
- **Icon-Only Buttons:** Icon sollte mindestens 20px sein für ausreichende Erkennbarkeit

#### Spacing:

- **Icon + Text:** 8px Abstand ( `icon.spacing.withText = spacing.2` )

```
<button>
  <icon width="20" /> <!-- 8px gap --> Text
</button>
```

- **Standalone Icons:** 4px optischer Rand ( `icon.spacing.standalone = spacing.1` )

### Code-Beispiele

#### React mit Lucide:

```

import { Search, Filter, Download } from 'lucide-react';

// Small (20px) - Button mit Text
<button>
  <Search size={20} />
  Suchen
</button>

// Medium (24px) - Icon-only Button
<button aria-label="Filter anzeigen">
  <Filter size={24} />
</button>

// Large (32px) - Feature-Highlight
<div className="feature">
  <Download size={32} />
  <h3>Dateien herunterladen</h3>
</div>

```

HTML mit SVG-Dateien:

```

<!-- Small (20px) - Inline Icon -->
<span class="icon" style="width: 20px; height: 20px;">
  <svg><!-- content from assets/icons/actions/search.svg --></svg>
</span>

<!-- Medium (24px) - Standalone -->
<button class="icon-button" aria-label="Menu">
  <svg width="24" height="24"><!-- content from navigation/menu.svg --></svg>
</button>

```

CSS mit Design Tokens:

```

.icon-xs { width: var(--icon-size-xs); height: var(--icon-size-xs); }
.icon-sm { width: var(--icon-size-sm); height: var(--icon-size-sm); }
.icon-md { width: var(--icon-size-md); height: var(--icon-size-md); }
.icon-lg { width: var(--icon-size-lg); height: var(--icon-size-lg); }
.icon-xl { width: var(--icon-size-xl); height: var(--icon-size-xl); }

.icon-with-text { gap: var(--icon-spacing-with-text); }

```

## Barrierefreiheit

Icons müssen WCAG 2.1 AA-konform verwendet werden.

### Dekorative Icons

Icons, die nur visuell unterstützen und von Text begleitet werden.

**Pattern:** `aria-hidden="true"` auf Icon-Element

```
<!-- Button mit Text UND Icon -->
<button>
  <svg aria-hidden="true" width="20" height="20"><!-- icon --></svg>
  Speichern
</button>
```

**Warum:** Screenreader würden sonst "Speichern-Grafik" oder Dateinamen vorlesen → redundant und verwirrend.

## Icon-Only Buttons

Interaktive Elemente ohne begleitenden Text.

**Pattern:** `aria-label` auf Button/Link-Element

```
<!-- Icon-Only Button -->
<button aria-label="Menu öffnen">
  <svg aria-hidden="true" width="24" height="24"><!-- menu icon --></svg>
</button>

<a href="/home" aria-label="Zur Startseite">
  <svg aria-hidden="true" width="24" height="24"><!-- home icon --></svg>
</a>
```

**Warum:** Screenreader brauchen Text-Alternative, um Funktion zu erklären.

## Informative Icons

Icons, die Information vermitteln ohne begleitenden Text.

**Pattern:** `role="img"` + `aria-label` auf SVG

```
<!-- Status-Icon ohne Text -->
<svg role="img" aria-label="Erfolg" width="24" height="24">
  <!-- check-circle icon -->
</svg>

<!-- Warnung ohne Text -->
<svg role="img" aria-label="Achtung: Diese Aktion kann nicht rückgängig gemacht werden" width="24" height="24">
  <!-- alert-triangle icon -->
</svg>
```

**Warum:** Icon selbst trägt die Information → muss von Screenreader kommuniziert werden.

## Touch Targets

**Minimum Touch Target:** 44×44px (WCAG 2.1 AAA)

- Token: `icon.touchTarget.minimum = 44px`
- Für alle interaktiven Icons auf mobilen Geräten

**Compact Touch Target:** 32×32px

- Token: `icon.touchTarget.compact = 32px`

- Nur für Desktop mit ausreichendem Abstand zu anderen Elementen

#### Implementation:

```
/* Mobile: Minimum 44px */
.icon-button {
  width: var(--icon-touch-target-minimum);
  height: var(--icon-touch-target-minimum);
  display: flex;
  align-items: center;
  justify-content: center;
}

/* Desktop: Kann kompakter sein wenn spacing stimmt */
@media (min-width: 1024px) {
  .icon-button.compact {
    width: var(--icon-touch-target-compact);
    height: var(--icon-touch-target-compact);
  }
}
```

#### Icon-Groesse vs Touch Target:

- Icon selbst kann klein sein (20px)
- Container muss Touch Target erfüllen (44px)
- Icon wird zentriert im Touch-Target-Container

## Export-Formate

### SVG (Primary)

Alle Icons sind als optimierte SVG-Dateien verfügbar:

- Pfad: design-system/assets/icons/
- Organisiert nach Kategorie: navigation/, actions/, status/, product/

#### SVG-Spezifikationen:

- viewBox: "0 0 24 24" (Standard für alle Icons)
- stroke: "currentColor" (erbt Textfarbe des Parent-Elements)
- stroke-width: "2"
- fill: "none"
- stroke-linecap: "round"
- stroke-linejoin: "round"
- aria-hidden: "true" (per Default, überschreiben wenn informativ)

#### Vorteile:

- Skaliert verlustfrei auf jede Größe
- currentColor ermöglicht einfaches Theming (Icon erbt Textfarbe)
- Kleine Dateigröße (durchschnittlich 250-400 Bytes nach Optimierung)
- Zugänglich über CSS fill/stroke

## Optimierung mit SVGO

**Config:** `design-system/svgo.config.js`

Alle exportierten SVGs sind optimiert:

- Entfernte Attribute: `class`, `data-*` (sauberer Code)
- Entfernte Dimensionen: `width`, `height` (nutzt viewBox für Skalierung)
- Hinzugefügt: `aria-hidden="true"` (Barrierefreiheit)
- Preset: `preset-default` (Standard-Optimierungen)

**Command:**

```
npx svgo -f assets/icons --config svgo.config.js -r
```

**Resultat:** 25-40% Größenreduktion, standardisierte Attribute

## PNG-Export (Fallback)

Für Legacy-Systeme oder Dokumentation können PNG-Exports generiert werden.

Nicht im Design-System enthalten, aber einfach zu generieren:

**Installation:**

```
npm install svgexport --save-dev
```

**2x Export (48px für Retina):**

```
npx svgexport assets/icons/actions/search.svg search@2x.png 48:48
```

**3x Export (72px für High-DPI):**

```
npx svgexport assets/icons/actions/search.svg search@3x.png 72:72
```

**Batch-Export aller Icons:**

```
find assets/icons -name "*.svg" -exec sh -c 'npx svgexport "$1" "${1%.svg}@2x.png" 48:48' {} \;
```

**Empfehlung:** Nur bei explizitem Bedarf exportieren. SVG ist in 99% der Fälle die bessere Wahl.

## Don'ts

### Nicht: Icons mit Text versehen

Falsch:

```
<svg>
  <path d="..." />
  <text>Download</text>
</svg>
```

**Warum:** Verhindert Lokalisierung, Text ist nicht stylebar, Barrierefreiheit problematisch.

**Richtig:** Text separat im HTML, Icon bleibt rein visuell.

### Nicht: Willkürliche Größen verwenden

Falsch:

```
<svg width="17" height="17">...</svg>
<svg width="22" height="22">...</svg>
```

**Warum:** Inkonsistent, Icons sind nicht für diese Größen optimiert, visuelle Unruhe.

**Richtig:** Nur Token-basierte Größen verwenden (16, 20, 24, 32, 48).

### Nicht: Farbwerte hardcoden

Falsch:

```
<svg stroke="#008BD2">...</svg>
```

**Warum:** Verhindert Theming, Icons können nicht auf Kontext reagieren (hover, disabled), Wartbarkeit leidet.

**Richtig:** `stroke="currentColor"` nutzen, Farbe über CSS steuern.

### Nicht: Fokus-Indikatoren weglassen

Falsch:

```
.icon-button:focus {
  outline: none; /* NEVER */
}
```

**Warum:** WCAG-Violation, Tastatur-Navigation unmöglich, unzugaenglich für Screenreader-Nutzer.

**Richtig:** Deutlichen Focus-Indicator bereitstellen:

```
.icon-button:focus-visible {
  outline: 2px solid var(--hydrophon-blau-500);
  outline-offset: 2px;
}
```

## Nicht: Icons unterschiedlicher Stile mischen

Falsch: Lucide-Icons (2px stroke, rounded) + Heroicons-Solid (filled) gemischt

Warum: Visuelle Inkonsistenz, wirkt unprofessionell.

Richtig: Konsistent Lucide-Icons verwenden, bei Custom-Icons denselben Stil befolgen.

---

## Ressourcen

- **Lucide Icon Library:** <https://lucide.dev>
  - **WCAG 2.1 Guidelines:** <https://www.w3.org/WAI/WCAG21/quickref/>
  - **Design Tokens:** `design-system/tokens/icons.json`
  - **Exported Icons:** `design-system/assets/icons/`
  - **SVGO Config:** `design-system/svgo.config.js`
- 

Letzte Aktualisierung: 2026-01-28 Version: 1.0

---

# Buttons

---

Buttons ermöglichen Nutzer\*innen, Aktionen auszulösen und wichtige Entscheidungen zu treffen. Das Hydrophon Button-System nutzt eine klare visuelle Hierarchie, um die Wichtigkeit von Aktionen zu kommunizieren.

**Anforderungen:** BTN-01 (Primary Button), BTN-02 (Secondary Button), BTN-03 (Tertiary Button), BTN-04 (Button-Größen), BTN-05 (Icon-Buttons)

---

## Übersicht

Das Button-System verwendet drei Varianten mit abnehmender visueller Gewichtung:

### Button-Hierarchie:

1. **Primary Button** — Höchste Priorität, eine Hauptaktion pro Bereich
2. **Secondary Button** — Alternative Aktionen, gleichrangig untereinander
3. **Tertiary/Ghost Button** — Niedrigste Priorität, konkurriert nicht um Aufmerksamkeit

### Wann welche Variante:

- **Primary:** Die wichtigste Aktion im aktuellen Kontext (z.B. "Speichern", "Absenden", "Kaufen")
- **Secondary:** Alternative oder zusätzliche Aktionen (z.B. "Abbrechen", "Zurück", "Mehr erfahren")
- **Tertiary:** Optionale Aktionen mit niedrigster Priorität (z.B. "Überspringen", "Später", "Details ein-/ausblenden")

**Grundregel:** Maximal ein Primary Button pro Ansicht oder Abschnitt. Multiple Secondary Buttons sind erlaubt, sofern sie gleichrangige Alternativen darstellen.

---

## Button-Varianten

### Primary Button (BTN-01)

#### Visueller Stil:

- Gefüllter Hintergrund in Hydrophon Blau (#008BD2)
- Weißer Text für maximalen Kontrast
- Kein sichtbarer Rahmen (transparent)
- Dezentler Farbwechsel bei Interaktion (keine Größenänderung)

**Verwendungszweck:** Die wichtigste Aktion im aktuellen Kontext. Primary Buttons signalisieren den empfohlenen oder erwarteten nächsten Schritt.

#### Beispiele:

- Formular absenden ("Speichern", "Senden")
- Kaufentscheidung bestätigen ("Jetzt kaufen", "In den Warenkorb")
- Workflow fortsetzen ("Weiter", "Abschließen")

**Verwendungsregel:** Nur ein Primary Button pro Ansicht oder logischem Abschnitt. Bei mehreren wichtigen Aktionen Priorität festlegen und andere als Secondary kennzeichnen.

## Zustände (Primary Button)

Zustand	Hintergrund	Text	Rahmen	Token
Default	hydrophon.blau.500	Weiβ	transparent	button.primary.background.default
Hover	hydrophon.blau.600	Weiβ	transparent	button.primary.background.hover
Active	hydrophon.blau.700	Weiβ	transparent	button.primary.background.active
Focus	hydrophon.blau.500	Weiβ	2px Ring blau.300, 2px Offset	button.focus.outlineColor
Disabled	neutral.200	neutral.400	transparent	button.primary.background.disabled

### CSS-Beispiel:

```
.button--primary {
  background-color: var(--button-primary-background-default);
  color: var(--button-primary-foreground-default);
  border: none;
}

.button--primary:hover:not(:disabled) {
  background-color: var(--button-primary-background-hover);
}

.button--primary:active:not(:disabled) {
  background-color: var(--button-primary-background-active);
}

.button--primary:focus-visible {
  outline: var(--button-focus-outline-width) solid var(--button-focus-outline-color);
  outline-offset: var(--button-focus-outline-offset);
}

.button--primary:disabled {
  background-color: var(--button-primary-background-disabled);
  color: var(--button-primary-foreground-disabled);
  cursor: not-allowed;
}
```

## Secondary Button (BTN-02)

### Visueller Stil:

- Transparenter Hintergrund (Outline-Stil)
- Rahmen und Text in Hydrophon Blau (#008BD2)
- Subtile Hintergrundfarbe bei Hover
- Gleiche Rahmenbreite wie Primary (2px)

**Verwendungszweck:** Alternative Aktionen, die gleichrangig zu anderen Secondary Buttons sind. Secondary Buttons bieten Optionen, ohne die Aufmerksamkeit vom Primary Button abzulenken.

### Beispiele:

- Abbrechen oder Zurück ("Abbrechen", "Zurück zur Übersicht")

- Zusätzliche Informationen ("Mehr erfahren", "Details anzeigen")
- Alternative Workflows ("Als Entwurf speichern")

**Verwendungsregel:** Mehrere Secondary Buttons sind erlaubt, wenn sie gleichrangige Alternativen darstellen. Immer in Kombination mit einem Primary Button verwenden, um klare Hierarchie zu etablieren.

#### Zustände (Secondary Button)

Zustand	Hintergrund	Text	Rahmen	Token
Default	transparent	hydrophon.blau.500	2px hydrophon.blau.500	button.secondary.border.default
Hover	hydrophon.blau.50	hydrophon.blau.600	2px hydrophon.blau.600	button.secondary.background.hover
Active	hydrophon.blau.100	hydrophon.blau.600	2px hydrophon.blau.600	button.secondary.background.active
Focus	transparent	hydrophon.blau.500	2px Ring blau.300, 2px Offset	button.focus.outlineColor
Disabled	transparent	neutral.400	2px neutral.300	button.secondary.border.disabled

CSS-Beispiel:

```
.button--secondary {
  background-color: var(--button-secondary-background-default);
  color: var(--button-secondary-foreground-default);
  border: var(--button-base-border-width) solid var(--button-secondary-border-default);
}

.button--secondary:hover:not(:disabled) {
  background-color: var(--button-secondary-background-hover);
  color: var(--button-secondary-foreground-hover);
  border-color: var(--button-secondary-border-hover);
}

.button--secondary:active:not(:disabled) {
  background-color: var(--button-secondary-background-active);
}

.button--secondary:focus-visible {
  outline: var(--button-focus-outline-width) solid var(--button-focus-outline-color);
  outline-offset: var(--button-focus-outline-offset);
}

.button--secondary:disabled {
  color: var(--button-secondary-foreground-disabled);
  border-color: var(--button-secondary-border-disabled);
  cursor: not-allowed;
}
```

---

#### Tertiary/Ghost Button (BTN-03)

Visueller Stil:

- Kein Hintergrund, kein Rahmen (nur Text)
- Text in Hydrophon Blau (#008BD2)
- Sehr subtiler Hintergrund bei Hover (neutral.100)
- Minimale visuelle Gewichtung

**Verwendungszweck:** Optionale Aktionen mit niedrigster Priorität, die nicht um Aufmerksamkeit konkurrieren sollen.

Tertiary Buttons sind unauffällig und werden oft für sekundäre Navigation oder Toggle-Funktionen verwendet.

#### Beispiele:

- Optionale Schritte überspringen ("Überspringen", "Später erinnern")
- Details ein-/ausklappen ("Details einblenden", "Weniger anzeigen")
- Sekundäre Navigation ("Zu den FAQs", "Hilfe anzeigen")

**Verwendungsregel:** Sparsam einsetzen. Nur für Aktionen, die nicht im Fokus stehen sollen. Nicht für wichtige oder häufig benötigte Aktionen verwenden.

#### Zustände (Tertiary/Ghost Button)

Zustand	Hintergrund	Text	Rahmen	Token
Default	transparent	hydrophon.blau.500	transparent	button.tertiary.foreground.default
Hover	neutral.100	hydrophon.blau.600	transparent	button.tertiary.background.hover
Active	neutral.200	hydrophon.blau.600	transparent	button.tertiary.background.active
Focus	transparent	hydrophon.blau.500	2px Ring blau.300, 2px Offset	button.focus.outlineColor
Disabled	transparent	neutral.400	transparent	button.tertiary.foreground.disabled

#### CSS-Beispiel:

```
.button--tertiary {
  background-color: var(--button-tertiary-background-default);
  color: var(--button-tertiary-foreground-default);
  border: none;
}

.button--tertiary:hover:not(:disabled) {
  background-color: var(--button-tertiary-background-hover);
  color: var(--button-tertiary-foreground-hover);
}

.button--tertiary:active:not(:disabled) {
  background-color: var(--button-tertiary-background-active);
}

.button--tertiary:focus-visible {
  outline: var(--button-focus-outline-width) solid var(--button-focus-outline-color);
  outline-offset: var(--button-focus-outline-offset);
}

.button--tertiary:disabled {
  color: var(--button-tertiary-foreground-disabled);
  cursor: not-allowed;
}
```

## Button-Größen (BTN-04)

Das System bietet drei Button-Größen für unterschiedliche Anwendungsfälle:

Größe	Höhe	Padding	Schriftgröße	Icon-Größe	Anwendung
Small	32px	6px 12px	14px	16px	Kompakte UIs, Tabellen, Toolbar
Medium	40px	8px 16px	16px	20px	Standard, Formulare, Dialoge
Large	48px	12px 24px	18px	20px	Hero CTAs, Marketing-Seiten, Landingpages

### Small (32px)

**Verwendung:** Kompakte Benutzeroberflächen, Tabellen-Aktionen, Toolbar-Buttons.

**Touch-Target-Hinweis:** Small Buttons (32px) erfüllen nicht die WCAG 2.1 Empfehlung von 44x44px Touch-Targets. Nur auf Desktop-Ansichten verwenden, auf mobilen Geräten Medium oder Large bevorzugen.

**Tokens:**

- button.size.small.minHeight : 32px
- button.size.small.paddingX : 12px
- button.size.small.paddingY : 6px
- button.size.small.fontSize : 14px
- button.size.small.iconSize : 16px
- button.size.small.gap : 6px

### Medium (40px) – Standard

**Verwendung:** Standard-Größe für die meisten Anwendungsfälle. Formulare, Dialoge, modale Fenster, Standard-Navigation.

**Touch-Target:** Medium Buttons (40px) kommen der WCAG-Empfehlung nahe, mit ausreichend Abstand zu benachbarten Elementen erfüllen sie die Anforderungen.

**Tokens:**

- button.size.medium.minHeight : 40px
- button.size.medium.paddingX : 16px
- button.size.medium.paddingY : 8px
- button.size.medium.fontSize : 16px
- button.size.medium.iconSize : 20px
- button.size.medium.gap : 8px

### Large (48px)

**Verwendung:** Hero-Bereiche, Call-to-Actions auf Landingpages, Marketing-Seiten, wichtige primäre Aktionen.

**Touch-Target:** Large Buttons (48px) übertreffen die WCAG-Empfehlung und sind für alle Geräte geeignet.

**Tokens:**

- button.size.large.minHeight : 48px
- button.size.large.paddingX : 24px
- button.size.large.paddingY : 12px
- button.size.large.fontSize : 18px

- button.size.large.iconSize : 20px
  - button.size.large.gap : 8px
- 

## Icon-Buttons (BTN-05)

Icons können Buttons zusätzlichen Kontext geben oder als alleiniges visuelles Element dienen.

### Icon + Text Buttons

Icons in Kombination mit Text erhöhen die Erkennbarkeit und können Aktionen verdeutlichen.

#### Icon-Position:

- **Links (Standard):** Icon vor dem Text für Aktionen wie "Herunterladen", "Hinzufügen"
- **Rechts:** Icon nach dem Text für Navigations-Aktionen wie "Weiter", "Externe Links"

**Icon-Größe:** Die Icon-Größe passt sich automatisch an die Button-Größe an:

- Small Button (32px): Icon 16px ( icon.size.xs )
- Medium Button (40px): Icon 20px ( icon.size.sm )
- Large Button (48px): Icon 20px ( icon.size.sm )

#### Abstand:

- Small Button: 6px zwischen Icon und Text ( spacing.1.5 )
- Medium/Large Button: 8px zwischen Icon und Text ( spacing.2 )

**Icon-Farbe:** Icons verwenden `currentColor`, um automatisch die Textfarbe zu übernehmen. Keine separate Farbdefinition notwendig.

#### HTML-Beispiel (Icon links):

```
<button class="button button--primary button--medium">
  <svg class="icon icon--sm" aria-hidden="true" width="20" height="20">
    <use href="#icon-check"></use>
  </svg>
  <span>Speichern</span>
</button>
```

#### HTML-Beispiel (Icon rechts):

```
<button class="button button--secondary button--medium">
  <span>Mehr erfahren</span>
  <svg class="icon icon--sm" aria-hidden="true" width="20" height="20">
    <use href="#icon-arrow-right"></use>
  </svg>
</button>
```

#### CSS für Icon + Text Layout:

```

.button {
  display: inline-flex;
  align-items: center;
  gap: var(--button-size-medium-gap); /* 8px für Medium */
}

.button--small {
  gap: var(--button-size-small-gap); /* 6px für Small */
}

.button .icon {
  color:currentColor; /* Automatische Farbvererbung */
}

```

## Icon-Only Buttons

Buttons, die nur ein Icon ohne Text enthalten, eignen sich für wiedererkennbare Aktionen in kompakten UIs.

**Wichtig:** Barrierefreiheit Icon-Only Buttons müssen ein `aria-label` Attribut enthalten, um für Screenreader verständlich zu sein.

### Touch-Target-Anforderungen:

- **Small (32px):** Nur auf Desktop mit ausreichend Abstand zu anderen Elementen
- **Medium (44px):** Erfüllt WCAG 2.1 AAA Touch-Target-Empfehlung (44x44px)
- **Large (48px):** Für große Hero-Bereiche oder wichtige CTAs

### Icon-Only Button-Dimensionen:

Größe	Button-Größe	Icon-Größe	Touch-Target	Token
Small	32x32px	20px (sm)	Desktop only	button.iconOnly.small.size
Medium	44x44px	24px (md)	WCAG AAA	button.iconOnly.medium.size
Large	48x48px	24px (md)	WCAG AAA	button.iconOnly.large.size

### HTML-Beispiel:

```

<!-- Medium Icon-Only Button (empfohlen für mobile) -->
<button class="button button--secondary button--icon-only button--medium"
       aria-label="Menü öffnen">
  <svg class="icon icon--md" aria-hidden="true" width="24" height="24">
    <use href="#icon-menu"></use>
  </svg>
</button>

<!-- Small Icon-Only Button (nur Desktop) -->
<button class="button button--tertiary button--icon-only button--small"
       aria-label="Schließen">
  <svg class="icon icon--sm" aria-hidden="true" width="20" height="20">
    <use href="#icon-x"></use>
  </svg>
</button>

```

### CSS für Icon-Only Buttons:

```

.button--icon-only {
  padding: 0;
  display: inline-flex;
  align-items: center;
  justify-content: center;
}

.button--icon-only.button--small {
  width: var(--button-icon-only-small-size); /* 32px */
  height: var(--button-icon-only-small-size);
}

.button--icon-only.button--medium {
  width: var(--button-icon-only-medium-size); /* 44px */
  height: var(--button-icon-only-medium-size);
}

.button--icon-only.button--large {
  width: var(--button-icon-only-large-size); /* 48px */
  height: var(--button-icon-only-large-size);
}

```

## Barrierefreiheit (WCAG 2.1 AA)

### Focus-Indikatoren

Alle Buttons müssen einen sichtbaren Focus-Indikator haben, der den WCAG 2.2 Standard "Focus Appearance" erfüllt.

#### Anforderungen:

- Mindestens 2px Outline-Breite
- Mindestens 2px Abstand (Offset) zum Button
- Kontrast von mindestens 3:1 zwischen Focus-Zustand und Unfokussiert

#### Tokens:

- button.focus.outlineWidth : 2px
- button.focus.outlineOffset : 2px
- button.focus.outlineColor : hydrophon.blau.300

#### CSS-Implementierung:

```

.button:focus-visible {
  outline: var(--button-focus-outline-width) solid var(--button-focus-outline-color);
  outline-offset: var(--button-focus-outline-offset);
}

/* Niemals Focus-Indikatoren entfernen */
.button:focus {
  /* Fokus-Styles MÜSSEN vorhanden sein */
}

```

**Wichtig:** Verwenden Sie `:focus-visible` für moderne Browser, um Focus-Indikatoren nur bei Tastatur-Navigation anzuzeigen. Niemals `:focus { outline: none; }` verwenden.

## Disabled-Zustände

Deaktivierte Buttons sollten das `disabled` Attribut verwenden, nicht `aria-disabled`.

**Grund:** Das `disabled` Attribut entfernt Buttons aus der Tab-Reihenfolge und verhindert Interaktionen nativ. `aria-disabled` erfordert zusätzliches JavaScript, um Interaktionen zu blockieren.

**HTML:**

```
<!-- Richtig -->
<button class="button button--primary" disabled>Speichern</button>

<!-- Falsch -->
<button class="button button--primary" aria-disabled="true">Speichern</button>
```

**Kontrast:** Deaktivierte Buttons sind von WCAG-Kontrast-Anforderungen ausgenommen. Die Token-Werte für Disabled-Zustände (`neutral.200`, `neutral.400`) erfüllen jedoch 3:1 Kontrast für bessere Lesbarkeit.

**Best Practice:** Erwägen Sie, deaktivierte Buttons vollständig auszublenden, wenn die Aktion niemals möglich ist. Zeigen Sie deaktivierte Buttons nur, wenn der Nutzer die Bedingungen für die Aktivierung verstehen muss.

## Keyboard-Navigation

Alle Buttons müssen über die Tastatur erreichbar und bedienbar sein.

**Anforderungen:**

- Fokussierbar über Tab-Taste
- Aktivierbar über Enter oder Leerzeichen
- Fokus-Reihenfolge folgt visueller Reihenfolge

**Standard-Button-Element:** Das native `<button>` Element erfüllt alle Keyboard-Anforderungen automatisch. Keine zusätzlichen ARIA-Attribute notwendig.

**Vermeiden:**

- `<div>` oder `<span>` als Button (erfordert `role="button"`, `tabindex="0"`, und JavaScript für Keyboard-Events)
- Links (`<a>`) als Buttons (Links navigieren, Buttons führen Aktionen aus)

## Touch-Targets

**WCAG 2.1 AAA Empfehlung:** Mindestens 44x44px Touch-Target-Größe für mobile Geräte.

**Button-Größen und Touch-Targets:**

- **Small (32px):** Nicht WCAG-konform, nur Desktop mit ausreichend Abstand (WCAG 2.2 erlaubt 24px mit Spacing-Ausnahme)
- **Medium (40px):** Nahe an WCAG-Empfehlung, mit Abstand zu benachbarten Elementen konform
- **Large (48px):** Übertrifft WCAG-Empfehlung

**Empfehlung:**

- Mobile: Medium (40px) oder Large (48px)
- Desktop: Alle Größen erlaubt, Small nur mit ausreichend Abstand

**Abstand zwischen Buttons:** Mindestens 8px Abstand zwischen benachbarten Buttons (verwenden Sie `spacing.2` Token) für Touch-Sicherheit.

## CSS-Implementierungsleitfaden

### Basis-Button-Struktur

```
.button {  
  /* Base Styles */  
  display: inline-flex;  
  align-items: center;  
  justify-content: center;  
  font-family: var(--font-family-base); /* Inter */  
  font-weight: var(--button-base-font-weight); /* 500 */  
  border-radius: var(--button-base-border-radius); /* 4px */  
  transition: var(--button-base-transition); /* 150ms ease-in-out */  
  cursor: pointer;  
  text-decoration: none;  
  line-height: 1;  
  
  /* Prevent text selection */  
  user-select: none;  
  -webkit-user-select: none;  
}  
  
.button:disabled {  
  cursor: not-allowed;  
  pointer-events: none; /* Verhindert Hover auf deaktivierten Buttons */  
}
```

## Varianten-Modifier

```

/* Primary Button */
.button--primary {
  background-color: var(--button-primary-background-default);
  color: var(--button-primary-foreground-default);
  border: none;
}

.button--primary:hover:not(:disabled) {
  background-color: var(--button-primary-background-hover);
}

.button--primary:active:not(:disabled) {
  background-color: var(--button-primary-background-active);
}

.button--primary:disabled {
  background-color: var(--button-primary-background-disabled);
  color: var(--button-primary-foreground-disabled);
}

/* Secondary Button */
.button--secondary {
  background-color: var(--button-secondary-background-default);
  color: var(--button-secondary-foreground-default);
  border: var(--button-base-border-width) solid var(--button-secondary-border-default);
}

.button--secondary:hover:not(:disabled) {
  background-color: var(--button-secondary-background-hover);
  color: var(--button-secondary-foreground-hover);
  border-color: var(--button-secondary-border-hover);
}

.button--secondary:active:not(:disabled) {
  background-color: var(--button-secondary-background-active);
}

.button--secondary:disabled {
  color: var(--button-secondary-foreground-disabled);
  border-color: var(--button-secondary-border-disabled);
}

/* Tertiary/Ghost Button */
.button--tertiary {
  background-color: var(--button-tertiary-background-default);
  color: var(--button-tertiary-foreground-default);
  border: none;
}

.button--tertiary:hover:not(:disabled) {
  background-color: var(--button-tertiary-background-hover);
  color: var(--button-tertiary-foreground-hover);
}

.button--tertiary:active:not(:disabled) {

```

```

background-color: var(--button-tertiary-background-active);
}

.button--tertiary:disabled {
  color: var(--button-tertiary-foreground-disabled);
}

```

## Größen-Modifier

```

.button--small {
  min-height: var(--button-size-small-min-height); /* 32px */
  padding: var(--button-size-small-padding-y) var(--button-size-small-padding-x); /* 6px 12px */
  font-size: var(--button-size-small-font-size); /* 14px */
  gap: var(--button-size-small-gap); /* 6px */
}

.button--medium {
  min-height: var(--button-size-medium-min-height); /* 40px */
  padding: var(--button-size-medium-padding-y) var(--button-size-medium-padding-x); /* 8px 16px */
  font-size: var(--button-size-medium-font-size); /* 16px */
  gap: var(--button-size-medium-gap); /* 8px */
}

.button--large {
  min-height: var(--button-size-large-min-height); /* 48px */
  padding: var(--button-size-large-padding-y) var(--button-size-large-padding-x); /* 12px 24px */
  font-size: var(--button-size-large-font-size); /* 18px */
  gap: var(--button-size-large-gap); /* 8px */
}

```

## Focus-Indikatoren

```

.button:focus-visible {
  outline: var(--button-focus-outline-width) solid var(--button-focus-outline-color);
  outline-offset: var(--button-focus-outline-offset);
}

/* Entfernen Sie :focus-visible Outline nur, wenn alternativer visueller Indikator vorhanden ist */

```

## Icon-Only Modifier

```
.button--icon-only {
  padding: 0;
  gap: 0;
}

.button--icon-only.button--small {
  width: var(--button-icon-only-small-size); /* 32px */
  height: var(--button-icon-only-small-size);
  min-height: unset;
}

.button--icon-only.button--medium {
  width: var(--button-icon-only-medium-size); /* 44px */
  height: var(--button-icon-only-medium-size);
  min-height: unset;
}

.button--icon-only.button--large {
  width: var(--button-icon-only-large-size); /* 48px */
  height: var(--button-icon-only-large-size);
  min-height: unset;
}
```

## Icon-Styling

```
.button .icon {
  color:currentColor; /* Automatische Farbvererbung */
  flex-shrink: 0; /* Verhindert Icon-Verzerrung */
}

.button--small .icon {
  width: var(--button-size-small-icon-size); /* 16px */
  height: var(--button-size-small-icon-size);
}

.button--medium .icon {
  width: var(--button-size-medium-icon-size); /* 20px */
  height: var(--button-size-medium-icon-size);
}

.button--large .icon {
  width: var(--button-size-large-icon-size); /* 20px */
  height: var(--button-size-large-icon-size);
}
```

## Don'ts – Was Sie vermeiden sollten

### ✗ Mehrere Primary Buttons pro Bereich

**Problem:** Verwirrt Nutzer über die wichtigste Aktion.

**Lösung:** Nur ein Primary Button pro Ansicht oder logischem Abschnitt. Andere Aktionen als Secondary kennzeichnen.

```
<!-- Falsch -->
<div class="actions">
  <button class="button button--primary">Speichern</button>
  <button class="button button--primary">Abbrechen</button> <!-- Beide Primary -->
</div>

<!-- Richtig -->
<div class="actions">
  <button class="button button--primary">Speichern</button>
  <button class="button button--secondary">Abbrechen</button>
</div>
```

### ✗ Buttons ohne erkennbare Grenzen (außer Tertiary)

**Problem:** Primary und Secondary Buttons müssen klar als interaktive Elemente erkennbar sein.

**Lösung:** Nur Tertiary Buttons dürfen keinen Hintergrund/Rahmen haben.

### ✗ Focus-Indikatoren entfernen

**Problem:** Tastatur-Nutzer können nicht erkennen, welcher Button fokussiert ist. WCAG-Verstoß.

**Lösung:** Niemals `:focus { outline: none; }` verwenden. Focus-Indikatoren sind Pflicht.

```
/* Niemals: */
.button:focus {
  outline: none; /* ✗ WCAG-Verstoß */
}

/* Immer: */
.button:focus-visible {
  outline: 2px solid var(--button-focus-outline-color);
  outline-offset: 2px;
}
```

### ✗ Zu kleine Touch-Targets auf Mobilgeräten

**Problem:** Small Buttons (32px) sind auf Touch-Geräten schwer zu treffen.

**Lösung:** Auf mobilen Geräten Medium (40px) oder Large (48px) verwenden.

```
/* Responsive Button-Größen */
@media (max-width: 768px) {
  .button--small {
    /* Small Buttons auf Mobile vermeiden oder mit Abstand kompensieren */
    min-height: var(--button-size-medium-min-height);
  }
}
```

## ✗ Animierte Größenänderungen bei Hover/Active

**Problem:** Größenänderungen bei Hover verschieben umliegende Elemente und wirken unruhig.

**Lösung:** Nur Farbänderungen bei Hover/Active. Keine `transform: scale()` oder Padding-Änderungen.

```
/* Falsch */
.button:hover {
  transform: scale(1.05); /* ✗ Verschiebt Layout */
}

/* Richtig */
.button:hover {
  background-color: var(--button-primary-background-hover); /* ✓ Nur Farbe */
}
```

## ✗ Farbwerte hardcoden

**Problem:** Inkonsistenz mit Design-Tokens, schwer wartbar.

**Lösung:** Immer Design-Tokens verwenden, niemals Hex-Codes direkt.

```
/* Falsch */
.button--primary {
  background-color: #008bd2; /* ✗ Hardcoded */
}

/* Richtig */
.button--primary {
  background-color: var(--button-primary-background-default); /* ✓ Token */
}
```

## ✗ Icon-Only Buttons ohne aria-label

**Problem:** Screenreader können Button-Funktion nicht vermitteln. WCAG-Verstoß.

**Lösung:** Immer `aria-label` für Icon-Only Buttons verwenden.

```
<!-- Falsch -->
<button class="button button--icon-only">
  <svg class="icon">...</svg> <!-- ✗ Keine Beschreibung -->
</button>

<!-- Richtig -->
<button class="button button--icon-only" aria-label="Menü öffnen">
  <svg class="icon" aria-hidden="true">...</svg> <!-- ✓ aria-label vorhanden -->
</button>
```

## Token-Referenz

Alle Button-Tokens sind in `design-system/tokens/buttons.json` definiert und werden als CSS Custom Properties in `build/css/variables.css` kompiliert.

### Verfügbare Token-Kategorien:

- `button.primary.*` — Primary Button Farben
- `button.secondary.*` — Secondary Button Farben
- `button.tertiary.*` — Tertiary Button Farben
- `button.size.{small|medium|large}.*` — Button-Größen
- `button.base.*` — Basis-Eigenschaften (borderRadius, borderWidth, fontWeight, transition)
- `button.focus.*` — Focus-Indikatoren
- `button.iconOnly.*` — Icon-Only Button-Dimensionen

### Verwendung:

```
.button--primary {
  background-color: var(--button-primary-background-default);
}
```

Letzte Aktualisierung: 2026-01-28 Version: 1.0 Status: Final

# Formular-Komponenten

---

Vollstaendiges Formularsystem fuer konsistente Dateneingabe mit WCAG 2.1 AA Compliance.

## Uebersicht

Dieses Kapitel dokumentiert alle Form-Komponenten des Hydrophon Design Systems. Die Komponenten folgen einem **Native-First-Ansatz** mit Fokus auf Accessibility, Performance und konsistenter User Experience.

## Schnellzugriff

Komponente	Beschreibung	Dokument
Text Input	Einzelige Texteingabe mit 3 Groessen und 6 Zustanden	<a href="#">text-input.md</a>
Textarea	Mehrzeilige Texteingabe mit vertikalem Resize	<a href="#">textarea.md</a>
Select	Dropdown-Auswahl mit Native-First-Empfehlung	<a href="#">select.md</a>
Checkbox	Mehrfachauswahl mit opacity:0 Native-Pattern	<a href="#">checkbox.md</a>
Radio Button	Exklusive Auswahl mit fieldset/legend Gruppierung	<a href="#">radio-button.md</a>
Labels & Helper	Beschriftungen, Pflichtfeld-Kennzeichnung, Helper Text	<a href="#">labels-helper-text.md</a>
Validierung	Progressive Validation, Error States, Fehlermeldungen	<a href="#">validation.md</a>

## Design-Prinzipien

### 1. Native-First Approach

Wir verwenden native HTML-Elemente wo immer moeglich:

- `<input>` , `<textarea>` , `<select>` statt Div-basierter Custom-Controls
- Bewahrung der Accessibility-Tree durch `opacity: 0` statt `display: none`
- Native Tastaturnavigation (Tab, Space, Pfeiltasten) bleibt erhalten
- Native Validierung via HTML5 `required` , `pattern` , etc.

#### Warum?

- Screenreader-Kompatibilitaet out-of-the-box
- Mobile OS Picker (iOS Datum-Picker, Android Tastatur)
- Weniger JavaScript-Code zu warten
- Bessere Performance

### 2. Konsistenz mit Button-System

Alle Form-Komponenten teilen die gleichen Hoehen wie das Button-System (Phase 2):

Groesse	Hoehe	Verwendung
Small	32px	Kompakte UIs, nur Desktop (unter WCAG Touch-Target)
Medium	40px	Standard fuer die meisten Formulare
Large	48px	Mobile-optimiert, CTAs, uebertrifft WCAG AAA

**Gemeinsame Token:**

- Border-Radius: 4px (konsistent mit Buttons)
- Focus-Indikatoren: 2px Outline + 2px/3px Ring
- Font-Family: Inter (konsistent mit Typography-System)

### 3. Accessibility First

Alle Form-Komponenten erfüllen **WCAG 2.1 AA**:

- **WCAG 3.3.1 Fehlererkennung:** Fehler werden automatisch erkannt und beschrieben
- **WCAG 3.3.2 Labels:** Alle Eingaben haben sichtbare, verknüpfte Labels
- **WCAG 3.3.3 Fehlervorschläge:** Fehlermeldungen enthalten Korrekturhinweise
- **WCAG 2.4.7 Fokus sichtbar:** 2px Outline mit ausreichend Kontrast
- **WCAG 1.4.1 Verwendung von Farbe:** Fehler nicht nur durch Farbe (+ Icon + Text)

**ARIA-Attribute:**

- `aria-required="true"` für Pflichtfelder
- `aria-invalid="true"` für fehlerhafte Felder
- `aria-describedby` für Helper Text und Error Messages
- `role="alert"` für Error Messages (sofortige Screenreader-Ankündigung)

### 4. Outline-Stil

Alle Eingabe-Komponenten verwenden **Outline-Style** (nicht Filled):

- **Weisser Hintergrund** in allen Zuständen (außer Disabled: `neutral.50`)
- **Sichtbare Border** in allen Zuständen (Farbe ändert sich)
- **Klarer Focus-State** mit verdickter Border + Ring

**Warum Outline?**

- Professioneller B2B-Look
- Besserer Kontrast für Text
- Konsistent mit Button-System (Secondary/Tertiary haben auch Border)
- Weniger visuelles Gewicht (weisser Hintergrund statt grauer)

## Groessen-System

### Input-Höhen

Token	Wert	Komponenten
<code>input.height.sm</code>	32px	Text Input, Select (nur Desktop)
<code>input.height.md</code>	40px	Text Input, Select, Textarea (Standard)
<code>input.height.lg</code>	48px	Text Input, Select, Textarea (Mobile-optimiert)

## Checkbox/Radio-Groessen

Token	Wert	Verwendung
checkbox.size.default	20px	Standard-Groesse Desktop & Mobile
checkbox.size.large	24px	Mobile-optimierte Interfaces
radio.size.default	20px	Standard-Groesse Desktop & Mobile
radio.size.large	24px	Mobile-optimierte Interfaces

### Touch-Target Compliance:

- Checkbox/Radio 20px + Label = 44px+ Touch-Target (WCAG AAA konform)
- Input 40px Hoehe = WCAG AA konform (24px Minimum)
- Input 48px Hoehe = WCAG AAA konform (44px Empfehlung)

## Zustaende

Alle Eingabe-Komponenten unterstuetzen folgende Zustaende:

Zustand	Beschreibung	Token-Suffix
Default	Standard-Zustand, keine Interaktion	.default
Hover	Maus-Hover (nicht Touch)	.hover
Focus	Tastatur- oder Maus-Fokus	.focus
Error	Validierungsfehler	.error
Success	Erfolgreich validiert (optional)	.success
Disabled	Deaktiviert, nicht editierbar	.disabled

### Zusaetliche Zustaende fuer Checkbox/Radio:

- Checked/Selected: Angekreuzt/Ausgewaehlt
- Checked + Hover: Angekreuzt + Maus-Hover
- Checked + Disabled: Angekreuzt + Deaktiviert

## Token-Namenskonvention

Alle Form-Tokens folgen einer konsistenten Hierarchie:

{component}.{element}.{property}.{state}

Beispiele:

input.field.border.focus	→ Hydrophon Blau 500
input.label.color.default	→ Neutral 700
input.error.color	→ Color Error
checkbox.background.checked	→ Hydrophon Blau 500
checkbox.border.hover	→ Neutral 400
radio.dot.color	→ Neutral White
radio.group.gap	→ 12px (spacing.3)

## Token-Kategorien

Kategorie	Beschreibung	Beispiele
input.field.*	Haupt-Eingabefeld (Background, Border, Text)	.border.default , .background.focus
input.label.*	Label-Styling	.color.default , .fontSize , .marginBottom
input.helper.*	Helper Text	.color , .fontSize , .marginTop
input.error.*	Error Message	.color , .fontSize , .marginTop
input.required.*	Pflichtfeld-Indikator	.color , .symbol
input.focus.*	Focus-Ring	.ring.color , .ring.width
checkbox.*	Checkbox-spezifische Tokens	.size.default , .background.checked
radio.*	Radio-spezifische Tokens	.dot.size , .group.gap
form.spacing.*	Layout-Spacing	.fieldToField , .groupToGroup

## Validierungs-Strategie

Alle Formulare verwenden **Progressive Validation** ("Belohne frueh, bestrafte spaet"):

- 1. Initiale Eingabe:** Keine Validierung waehrend User tippt
- 2. Blur:** Erste Validierung wenn Feld verlassen wird
- 3. Nach Fehler:** Echtzeit-Validierung bei jeder Aenderung
- 4. Bei Korrektur:** Fehler verschwindet sofort

```
const form = useForm({
  mode: 'onBlur',           // Initiale Validierung bei blur
  reValidateMode: 'onChange' // Nach Fehler: live re-validieren
});
```

Siehe [validation.md](#) fuer Details.

## Installation

Die Form-Tokens sind Teil des Design-System-Builds.

### Build ausfuehren

```
cd design-system
npm run build
```

#### Generierte Dateien:

- build/css/variables.css - CSS Custom Properties
- build/json/tokens.json - JSON Token-Export
- build/scss/\_variables.scss - SCSS Variablen (falls konfiguriert)

## Verwendung in CSS

```
@import '../design-system/build/css/variables.css';

.my-input {
  height: var(--input-height-md);
  border: 1px solid var(--input-field-border-default);
  border-radius: var(--input-base-border-radius);
  padding: 0 var(--input-padding-x-md);
  font-size: var(--input-font-size-md);
}

.my-input:focus {
  border-color: var(--input-field-border-focus);
  outline: var(--input-focus-ring-width) solid var(--input-focus-ring-color);
}

.my-input.error {
  border-color: var(--input-field-border-error);
}
```

## Verwendung in JavaScript/TypeScript

```
import tokens from '../design-system/build/json/tokens.json';

const inputHeight = tokens.input.height.md.$value; // "40px"
const errorColor = tokens.input.error.color.$value; // "#..."
```

## Code-Beispiele

### Vollstaendiges Formular (React Hook Form + Zod)

Siehe [validation.md](#) fuer ein vollstaendiges Integrations-Beispiel mit:

- React Hook Form Setup
- Zod Schema Validierung
- Progressive Validation
- Error/Success States
- Accessibility-Attribute (aria-invalid, aria-describedby, role="alert")
- CSS mit Token-Referenzen

## Einfaches Textfeld

```
<div class="form-field">
  <label for="email">
    E-Mail-Adresse
    <span class="required" aria-label="Pflichtfeld">*</span>
  </label>
  <input
    type="email"
    id="email"
    required
    aria-required="true"
    aria-describedby="email-helper"
  />
  <span id="email-helper" class="helper-text">
    Wir verwenden deine E-Mail nur zur Kontaktaufnahme
  </span>
</div>
```

## Checkbox-Gruppe

```
<fieldset class="form-group">
  <legend>Newsletter-Einstellungen</legend>

  <label class="checkbox-label">
    <input type="checkbox" name="newsletter" value="products" class="checkbox-native" />
    <span class="checkbox-custom"></span>
    Produktneuheiten
  </label>

  <label class="checkbox-label">
    <input type="checkbox" name="newsletter" value="updates" class="checkbox-native" />
    <span class="checkbox-custom"></span>
    System-Updates
  </label>
</fieldset>
```

## Radio-Gruppe

```
<fieldset class="form-group">
  <legend>
    Versandart
    <span class="required" aria-label="Pflichtfeld">*</span>
  </legend>

  <label class="radio-label">
    <input type="radio" name="shipping" value="standard" class="radio-native" required />
    <span class="radio-custom"></span>
    Standard (3-5 Tage)
  </label>

  <label class="radio-label">
    <input type="radio" name="shipping" value="express" class="radio-native" required />
    <span class="radio-custom"></span>
    Express (1-2 Tage)
  </label>
</fieldset>
```

## Best Practices

### Labels

- Immer ein sichtbares Label (nicht nur Placeholder)
- Label UEBER dem Input (nicht links, nicht rechts)
- for/id Verknuepfung fuer maximale Kompatibilitaet
- Kurz und praegnant (max 3-4 Woerter)

### Pflichtfelder

- Rotes Asterisk nach dem Label mit `aria-label="Pflichtfeld"`
- `required + aria-required="true"` auf dem Input
- Legende am Formularanfang ("Felder mit \* sind Pflichtfelder")
- Alternativ: Bei 80%+ Pflichtfeldern → "(optional)" bei optionalen Feldern

### Helper Text

- Unter dem Input mit `aria-describedby`
- Bleibt sichtbar (im Gegensatz zu Placeholder)
- Erklaert Format/Anforderungen BEVOR User eingibt
- Wird ersetzt durch Error-Message bei Fehler (oder bleibt bei kritischen Infos)

### Fehlermeldungen

- Erklaerend-hilfreich: Was ist falsch + Wie beheben + Beispiel
- Drei visuelle Indikatoren: Farbe + Icon + Text (WCAG 1.4.1)
- `role="alert"` fuer sofortige Screenreader-Ankuendigung
- Inline + Summary bei komplexen Formularen (5+ Felder)

## Tastatur-Navigation

- **Tab:** Nächstes Feld
- **Shift+Tab:** Vorheriges Feld
- **Space:** Checkbox/Radio toggle, Buttons aktivieren
- **Pfeiltasten:** Radio-Gruppe navigieren (native Verhalten)
- **Enter:** Submit-Button aktivieren

## Weiterführende Ressourcen

### Interne Dokumentation

- [Button-Dokumentation](#) - Button-System (Phase 2)
- [Icon-System](#) - Lucide Icons Integration
- [Farb-Tokens](#) - Color-System (Phase 1)
- [Typography](#) - Typography-System (Phase 1)

### Externe Referenzen

- [WCAG 2.1 Guidelines](#)
- [MDN Web Forms Guide](#)
- [React Hook Form Documentation](#)
- [Zod Validation Library](#)

---

Phase: 03 - Forms & Data Input Zuletzt aktualisiert: 2026-01-29 Version: 1.0

---

# Text Input

---

Einzeiliges Texteingabefeld für kurze Eingaben wie Name, E-Mail, Telefonnummer oder andere strukturierte Daten.

## Übersicht

Text Inputs sind die grundlegendsten und am häufigsten verwendeten Formularelemente. Sie ermöglichen Benutzern die Eingabe von kurzen, einzeiligen Texten und bieten visuelles Feedback über den aktuellen Zustand (Standard, Hover, Focus, Error, Success, Disabled).

## Anatomie

Ein vollständiges Text Input besteht aus mehreren Komponenten:

- **Container:** Border + Background bilden den sichtbaren Rahmen
- **Input-Text:** Der vom Benutzer eingegebene Text
- **Label:** Beschriftung oberhalb des Inputs (Pflicht für Accessibility)
- **Placeholder:** Optional - Beispieltext innerhalb des Inputs (nur als Ergänzung, nicht als Ersatz für Label)
- **Helper Text:** Optional - Zusätzliche Erklärungen unterhalb des Inputs
- **Error Message:** Fehlermeldung unterhalb bei ungültiger Eingabe
- **Success Icon:** Grünes Checkmark rechts im Input bei gültiger Eingabe
- **Required Indicator:** Rotes Sternchen (\*) bei Pflichtfeldern

## Verwendung

Text Inputs werden verwendet für:

- **Formulare aller Art:** Kontaktformulare, Registrierung, Checkout-Prozesse
- **Suchfelder:** Mit zusätzlichem Such-Icon (Lucide Search Icon, 20px)
- **Filter-Eingaben:** In Tabellen und Listen zur Dateneingabe
- **Strukturierte Daten:** Name, E-Mail, Telefon, PLZ, etc.

**Wann nicht verwenden:**

- Mehrzeilige Texte → Textarea verwenden
- Auswahl aus vordefinierten Optionen → Select/Dropdown verwenden
- Datum/Zeit → Date Picker verwenden (separate Phase)
- Ja/Nein Entscheidungen → Checkbox/Radio Button verwenden

## Größen

Das Text Input System bietet drei Größen, die mit dem Button-System (Phase 2) harmonieren.

### Small (32px)

**Spezifikationen:**

- Höhe: 32px (input.height.sm)
- Padding horizontal: 8px (input.padding.x.sm)

- Schriftgröße: 14px (input.font.size.sm)
- Icon-Größe: 16px (icon.size.xs)

**Verwendung:**

- Kompakte Desktop-Interfaces mit begrenztem Platz
- Inline-Editing in Tabellen
- Admin-Panels mit hoher Informationsdichte

**Wichtiger Hinweis:** Small Inputs sind **nur für Desktop** geeignet, da die 32px Höhe unter der WCAG-empfohlenen Touch-Target-Mindestgröße von 44px liegt. Auf mobilen Geräten immer Medium oder Large verwenden.

**Medium (40px) - Standard****Spezifikationen:**

- Höhe: 40px (input.height.md)
- Padding horizontal: 12px (input.padding.x.md)
- Schriftgröße: 16px (input.font.size.md)
- Icon-Größe: 20px (icon.size.sm)

**Verwendung:**

- Standard für die meisten Formulare
- Ausgewogenes Verhältnis zwischen Kompaktheit und Benutzerfreundlichkeit
- Empfohlen für responsive Layouts (Desktop + Mobile)

**Vorteile:**

- 16px Schriftgröße verhindert Zoom auf iOS Safari
- 40px Höhe bietet gute Touch-Ergonomie
- Konsistent mit Medium Buttons

**Large (48px)****Spezifikationen:**

- Höhe: 48px (input.height.lg)
- Padding horizontal: 16px (input.padding.x.lg)
- Schriftgröße: 18px (input.font.size.lg)
- Icon-Größe: 20px (icon.size.sm)

**Verwendung:**

- Mobile-optimierte Formulare
- Hero-Sections mit Suchfeld oder Newsletter-Anmeldung
- Call-to-Action Formulare (z.B. "Produksuche starten")
- Ältere Zielgruppen oder barrierefreie Anwendungen

**Vorteile:**

- Übertrifft WCAG AAA Touch-Target-Empfehlung (44px)
- Großzügige Klickfläche reduziert Fehleingaben
- Gute Lesbarkeit durch 18px Schriftgröße

**Zustände**

Text Inputs durchlaufen verschiedene Zustände während der Benutzerinteraktion. Jeder Zustand hat eine klare visuelle Differenzierung.

## Default (Standard)

Der Ausgangszustand eines leeren oder ausgefüllten Inputs ohne Benutzerinteraktion.

### Visuelle Eigenschaften:

- Border: `1px solid neutral.300` (hellgrau)
- Background: `neutral.white` (weiß)
- Text: `neutral.900` (dunkelgrau, hoher Kontrast)
- Placeholder: `neutral.400` (gedämpftes grau)

### Design Tokens:

```
input.field.background.default
input.field.border.default
input.field.borderWidth.default
input.field.text.default
input.field.text.placeholder
```

## Hover

Zustand wenn der Mauszeiger über dem Input schwebt (Desktop only).

### Visuelle Eigenschaften:

- Border: `1px solid neutral.400` (etwas dunkler als default)
- Background: `neutral.white` (unverändert)
- Cursor: `text` (I-Beam Cursor)

### Design Tokens:

```
input.field.background.hover
input.field.border.hover
```

**Wichtig:** Hover-Zustand gibt subtile visuelles Feedback, dass das Element interaktiv ist. Der Unterschied zum Default-Zustand ist bewusst dezent gehalten, um nicht mit dem Focus-Zustand zu konkurrieren.

## Focus

Der wichtigste Zustand - zeigt an, dass das Input-Feld aktiv ist und Tastatureingaben empfängt.

### Visuelle Eigenschaften:

- Border: `2px solid hydrophon.blau.500` (Hydrophon Blau, verstärkt auf 2px)
- Focus Ring: `3px hydrophon.blau.100` (heller Blauton als äußerer Ring)
- Background: `neutral.white` (unverändert)
- Outline Offset: `0px` (direkt am Input-Rand)

### Design Tokens:

```
input.field.background.focus
input.field.border.focus
input.field.borderWidth.focus
input.focus.ring.color
input.focus.ring.width
input.focus.outline.offset
```

**WCAG 2.4.7 Konformität:**

- **Sichtbarkeit:** 2px Border + 3px Ring = 5px Gesamtdicke, deutlich sichtbar
- **Kontrast:** Hydrophon Blau 500 zu weißem Hintergrund > 4.5:1 Kontrast
- **:focus-visible verwenden:** Nur bei Tastatur-Navigation anzeigen, nicht bei Maus-Klick

**Implementierungshinweis:**

```
input:focus-visible {
  outline: 2px solid var(--input-field-border-focus);
  outline-offset: 0px;
  box-shadow: 0 0 0 3px var(--input-focus-ring-color);
}
```

**Error**

Zustand bei ungültiger Eingabe nach Validierung.

**Visuelle Eigenschaften:**

- Border: 1px solid color.error (rot)
- Background: neutral.white (unverändert)
- Error Icon: Lucide AlertCircle (20px) rechts im Input, rot
- Error Message: Unterhalb des Inputs, 12px, rot
- Label: Optional in color.error färben

**Design Tokens:**

```
input.field.background.error
input.field.border.error
input.error.color
input.error.fontSize
input.error.marginTop
input.label.color.error
```

**WCAG 3.3.1 / 3.3.3 Konformität:**

- Fehler werden visuell (roter Border + Icon) UND textuell (Error Message) kommuniziert
- Error Message ist erklärend und hilfreich, nicht nur "ungültig"
- Fehlervorschläge mit Beispielen werden gegeben

**Beispiel Error Messages:**

- Schlecht: "Ungültige Eingabe"
- Gut: "Bitte gib eine gültige E-Mail-Adresse ein (z.B. name@firma.de)"
- Gut: "Die Telefonnummer muss mindestens 10 Ziffern enthalten"

**Accessibility Attribute:**

```

<input
  type="email"
  id="email"
  aria-invalid="true"
  aria-describedby="email-error"
/>

  Bitte gib eine gültige E-Mail-Adresse ein


```

## Success

Zustand bei gültiger Eingabe - positives Feedback für korrekte Daten.

### Visuelle Eigenschaften:

- Border: 1px solid color.success (grün)
- Background: neutral.white (unverändert)
- Success Icon: Lucide Check oder CheckCircle (20px) rechts im Input, grün
- Optional: Subtiler grüner Text unterhalb "E-Mail-Adresse ist gültig"

### Design Tokens:

```

input.field.background.success
input.field.border.success
input.success.color

```

### Verwendung:

- E-Mail-Validierung (Format-Prüfung)
- Passwort-Stärke (bei Erfüllung aller Kriterien)
- Verfügbarkeits-Prüfung (z.B. Benutzername verfügbar)

**Wichtig:** Success State nicht übermäßig verwenden - nur bei wichtigen Validierungen, nicht bei jedem Feld. Zu viel grün kann überladen wirken.

## Disabled

Zustand für nicht-editierbare Felder, die aus kontextuellen Gründen deaktiviert sind.

### Visuelle Eigenschaften:

- Background: neutral.50 (leichtes Grau)
- Border: 1px solid neutral.200 (helles Grau)
- Text: neutral.600 (reduziert, aber lesbar)
- Opacity: 0.7 (subtile Reduktion)
- Cursor: not-allowed

### Design Tokens:

```

input.field.background.disabled
input.field.border.disabled
input.field.text.disabled

```

### Accessibility:

- `aria-disabled="true"` setzen
- Alternativ: `disabled` Attribut (verhindert Fokus)
- Wichtig: Text bleibt lesbar (neutral.600), nicht zu stark abgeschwächt
- Tooltip mit Erklärung warum Feld deaktiviert ist (empfohlen)

#### Verwendung:

- Abhängige Felder (z.B. "Lieferadresse" wenn "Wie Rechnungsadresse" aktiv)
- Automatisch gefüllte Felder (z.B. berechnete Werte)
- Felder ohne Berechtigung (z.B. Admin-only Felder)

## Validierung

Validierung ist entscheidend für gute Formular-UX. Das Timing und die Art des Feedbacks beeinflussen die Benutzererfahrung massiv.

### Progressive Validation (Empfohlene Strategie)

#### Phase 1 - Initial (Erstes Ausfüllen):

- Validierung erfolgt bei `onBlur` (Feld verlassen)
- Verhindert irritierende Fehlermeldungen während der Eingabe
- Benutzer kann in Ruhe tippen ohne sofortige Kritik

#### Phase 2 - Nach Fehler:

- Validierung wechselt zu `onChange` (jeder Tastendruck)
- Benutzer bekommt sofortiges Feedback während Korrektur
- Fehler verschwindet sofort bei korrekter Eingabe

#### Phase 3 - Nach Erfolg:

- Zurück zu `onBlur` Validierung
- Reduziert Ablenkung durch ständige Success-Indikatoren

#### Vorteile dieser Strategie:

- Beste Balance zwischen Hilfestellung und Nicht-Stören
- Unterstützt von UX-Studien (Nielsen Norman Group)
- Reduziert kognitive Belastung

## Validierungstypen

#### Format-Validierung:

- E-Mail: Regex-Pattern für valides Format
- Telefon: Länge, erlaubte Zeichen, optional Länderformat
- URL: Protokoll, Domain-Struktur
- PLZ: Länderabhängige Formate

#### Wertebereichs-Validierung:

- Minimale/Maximale Länge (z.B. Passwort mindestens 8 Zeichen)
- Numerische Bereiche (z.B. Alter zwischen 18-99)
- Pflichtfeld-Prüfung

#### Kontextuelle Validierung:

- Verfügbarkeit (z.B. Benutzername bereits vergeben)
- Konsistenz (z.B. "Passwort wiederholen" muss übereinstimmen)

- Business-Logik (z.B. Enddatum nach Startdatum)

## Fehlermeldungen Best Practices

Struktur einer guten Fehlermeldung:

1. **Problem:** Was ist falsch?
2. **Ursache:** Warum ist es falsch? (optional)
3. **Lösung:** Wie kann es behoben werden? (mit Beispiel)

Beispiele:

- |  |
|--|
| <input checked="" type="checkbox"/> Schlecht: "Ungültig"<br><input checked="" type="checkbox"/> Gut: "Bitte gib eine gültige E-Mail-Adresse ein (z.B. name@firma.de)"            |
| <input checked="" type="checkbox"/> Schlecht: "Falsche Länge"<br><input checked="" type="checkbox"/> Gut: "Das Passwort muss mindestens 8 Zeichen lang sein"                     |
| <input checked="" type="checkbox"/> Schlecht: "Error 422"<br><input checked="" type="checkbox"/> Gut: "Diese E-Mail-Adresse ist bereits registriert. Hast du bereits ein Konto?" |

Ton:

- Freundlich und hilfreich, nicht anklagend
- Konkret, nicht vage
- Mit Beispielen, wenn möglich
- Deutsche Sprache, klar und präzise

## Accessibility (WCAG 2.1 AA)

Barrierefreiheit ist kein Optional - jedes Input muss diese Standards erfüllen.

### WCAG Anforderungen

#### 3.3.1 Fehlererkennung (Level A):

- Fehler werden automatisch erkannt
- Fehler werden dem Benutzer in Textform beschrieben
- Implementierung: Error Message unterhalb + `aria-invalid="true"`

#### 3.3.2 Labels oder Anweisungen (Level A):

- Jedes Input hat ein sichtbares Label
- Label ist mit Input verknüpft (`for / id`)
- Pflichtfelder sind gekennzeichnet (Required Indicator)

#### 3.3.3 Fehlervorschläge (Level AA):

- Korrekturvorschläge werden gegeben
- Implementierung: Error Messages mit Beispielen und Lösungen

#### 2.4.7 Fokus sichtbar (Level AA):

- Focus-Indikator ist deutlich sichtbar
- Minimum 2px Outline-Breite
- Minimum 3:1 Kontrast zum Hintergrund
- Implementierung: 2px Border + 3px Ring (gesamt 5px)

#### 4.1.2 Name, Rolle, Wert (Level A):

- Alle Formularelemente haben korrekte ARIA-Attribute
- Status-Änderungen werden assistiven Technologien kommuniziert

### HTML-Struktur mit ARIA

Minimale Struktur:

```
<div class="form-field">
  <label for="email" class="input-label">
    E-Mail-Adresse <span class="required" aria-label="Pflichtfeld">*</span>
  </label>
  <input
    type="email"
    id="email"
    name="email"
    class="text-input"
    placeholder="name@firma.de"
    required
    aria-required="true"
  />
</div>
```

Mit Helper Text:

```
<div class="form-field">
  <label for="password">Passwort <span class="required">*</span></label>
  <input
    type="password"
    id="password"
    aria-describedby="password-help"
    aria-required="true"
  />
  <span id="password-help" class="helper-text">
    Mindestens 8 Zeichen, 1 Großbuchstabe, 1 Zahl
  </span>
</div>
```

Mit Error State:

```

<div class="form-field error">
  <label for="email">E-Mail-Adresse <span class="required">*</span></label>
  <input
    type="email"
    id="email"
    aria-invalid="true"
    aria-describedby="email-error"
  />
  <span id="email-error" class="error-text" role="alert">
    Bitte gib eine gültige E-Mail-Adresse ein (z.B. name@firma.de)
  </span>
</div>

```

## Tastatur-Navigation

### Fokus-Reihenfolge:

- Tab: Zum nächsten Input springen
- Shift + Tab: Zum vorherigen Input springen
- Enter: Formular absenden (bei letztem Input)

### Focus Management:

- `:focus-visible` verwenden (nicht `:focus`)
- Verhindert Focus-Ring bei Maus-Klick, zeigt ihn bei Tastatur-Navigation
- Browser-Standard-Fokus nicht entfernen (`outline: none` vermeiden)

## Screen Reader Unterstützung

### Ansagen:

- Label wird vorgelesen: "E-Mail-Adresse, Pflichtfeld, Eingabefeld"
- Helper Text wird vorgelesen (via `aria-describedby` )
- Error Message wird vorgelesen (via `role="alert"` )
- Required State wird erkannt (via `aria-required="true"` )

### Best Practices:

- Placeholder ist KEIN Ersatz für Label (verschwindet bei Eingabe)
- Visuelles Label muss mit programmatischem Label übereinstimmen
- Error Icon allein reicht nicht - Text ist Pflicht

## Code-Beispiel

### CSS mit Design Tokens

```

/* Text Input Base Styles */
.text-input {
  /* Sizing */
  height: var(--input-height-md); /* 40px */
  padding: 0 var(--input-padding-x-md); /* 0 12px */

  /* Typography */
  font-family: var(--input-base-font-family);
  font-size: var(--input-font-size-md); /* 16px */
  color: var(--input-field-text-default);

  /* Visual */
  background-color: var(--input-field-background-default);
  border: var(--input-field-border-width-default) solid var(--input-field-border-default);
  border-radius: var(--input-base-border-radius); /* 4px */

  /* Interaction */
  transition: var(--input-base-transition); /* 150ms ease-in-out */
  cursor: text;
}

/* Placeholder */
.text-input::placeholder {
  color: var(--input-field-text-placeholder);
  opacity: 1; /* Firefox fix */
}

/* Hover State */
.text-input:hover:not(:disabled) {
  border-color: var(--input-field-border-hover);
}

/* Focus State (WCAG 2.2) */
.text-input:focus-visible {
  outline: var(--input-field-border-width-focus) solid var(--input-field-border-focus);
  outline-offset: var(--input-focus-outline-offset);
  box-shadow: 0 0 0 var(--input-focus-ring-width) var(--input-focus-ring-color);
  border-color: var(--input-field-border-focus);
}

/* Error State */
.text-input[aria-invalid="true"] {
  border-color: var(--input-field-border-error);
}

/* Success State */
.text-input.success {
  border-color: var(--input-field-border-success);
}

/* Disabled State */
.text-input:disabled {
}

```

```

background-color: var(--input-field-background-disabled);
border-color: var(--input-field-border-disabled);
color: var(--input-field-text-disabled);
cursor: not-allowed;
opacity: 0.7;
}

/* Label */
.input-label {
  display: block;
  font-size: var(--input-label-font-size); /* 14px */
  font-weight: var(--input-label-font-weight); /* 500 */
  color: var(--input-label-color-default);
  margin-bottom: var(--input-label-margin-bottom); /* 8px */
}

/* Required Indicator */
.required {
  color: var(--input-required-color);
  margin-left: 2px;
}

/* Helper Text */
.helper-text {
  display: block;
  font-size: var(--input-helper-font-size); /* 12px */
  color: var(--input-helper-color);
  margin-top: var(--input-helper-margin-top); /* 4px */
}

/* Error Message */
.error-text {
  display: block;
  font-size: var(--input-error-font-size); /* 12px */
  color: var(--input-error-color);
  margin-top: var(--input-error-margin-top); /* 4px */
}

/* Size Variants */
.text-input--small {
  height: var(--input-height-sm); /* 32px */
  padding: 0 var(--input-padding-x-sm); /* 0 8px */
  font-size: var(--input-font-size-sm); /* 14px */
}

.text-input--large {
  height: var(--input-height-lg); /* 48px */
  padding: 0 var(--input-padding-x-lg); /* 0 16px */
  font-size: var(--input-font-size-lg); /* 18px */
}

```

## HTML Beispiele

Standard Input mit Label:

```
<div class="form-field">
  <label for="name" class="input-label">Name</label>
  <input
    type="text"
    id="name"
    name="name"
    class="text-input"
    placeholder="Max Mustermann"
  />
</div>
```

Required Input mit Helper Text:

```
<div class="form-field">
  <label for="email" class="input-label">
    E-Mail-Adresse <span class="required">*</span>
  </label>
  <input
    type="email"
    id="email"
    name="email"
    class="text-input"
    placeholder="name@firma.de"
    required
    aria-required="true"
    aria-describedby="email-help"
  />
  <span id="email-help" class="helper-text">
    Wir senden dir eine Bestätigungs-E-Mail
  </span>
</div>
```

Error State:

```
<div class="form-field">
  <label for="phone" class="input-label">Telefonnummer <span class="required">*</span></label>
  <input
    type="tel"
    id="phone"
    name="phone"
    class="text-input"
    value="123"
    aria-invalid="true"
    aria-describedby="phone-error"
  />
  <span id="phone-error" class="error-text" role="alert">
    Die Telefonnummer muss mindestens 10 Ziffern enthalten
  </span>
</div>
```

## Design Tokens Referenz

Token Name	Wert	Verwendung
input.height.sm	32px	Input-Höhe Small
input.height.md	40px	Input-Höhe Medium (Standard)
input.height.lg	48px	Input-Höhe Large
input.padding.x.sm	8px	Horizontales Padding Small
input.padding.x.md	12px	Horizontales Padding Medium
input.padding.x.lg	16px	Horizontales Padding Large
input.font.size.sm	14px	Schriftgröße Small
input.font.size.md	16px	Schriftgröße Medium
input.font.size.lg	18px	Schriftgröße Large
input.field.background.default	neutral.white	Hintergrund Standard
input.field.background.disabled	neutral.50	Hintergrund Disabled
input.field.border.default	neutral.300	Border Standard
input.field.border.hover	neutral.400	Border Hover
input.field.border.focus	hydrophon.blau.500	Border Focus
input.field.border.error	color.error	Border Error
input.field.border.success	color.success	Border Success
input.field.border.disabled	neutral.200	Border Disabled
input.field.borderWidth.default	1px	Border-Breite Standard
input.field.borderWidth.focus	2px	Border-Breite Focus
input.field.text.default	neutral.900	Text-Farbe Standard
input.field.text.placeholder	neutral.400	Placeholder-Farbe
input.field.text.disabled	neutral.600	Text-Farbe Disabled
input.focus.ring.color	hydrophon.blau.100	Focus Ring Farbe
input.focus.ring.width	3px	Focus Ring Breite
input.label.color.default	neutral.700	Label-Farbe
input.label.fontSize	14px	Label-Schriftgröße
input.label.fontWeight	500	Label-Schriftgewicht
input.label.marginBottom	8px	Label Abstand zum Input
input.helper.color	neutral.600	Helper Text Farbe
input.helper.fontSize	12px	Helper Text Schriftgröße
input.error.color	color.error	Error Text Farbe
input.error.fontSize	12px	Error Text Schriftgröße
input.base.borderRadius	4px	Eckenradius
input.base.transition	150ms ease-in-out	Übergangsanimation

**Version:** 1.0 **Letzte Aktualisierung:** 2026-01-29 **Phase:** 03 - Forms & Data Input **Komponente:** FORM-01 - Text Input

---

# Textarea

---

Mehrzeiliges Texteingabefeld für längere Textinhalte wie Kommentare, Beschreibungen, Nachrichten oder Notizen.

## Übersicht

Textarea-Komponenten ermöglichen Benutzern die Eingabe von mehrzeiligen Texten mit variabler Länge. Im Gegensatz zu einzeiligen Text Inputs bieten Textareas mehr Platz und visuelle Hinweise darauf, dass längere Inhalte erwartet werden.

## Anatomie

Eine vollständige Textarea besteht aus:

- **Container:** Border + Background mit variabler Höhe
- **Input-Text:** Mehrzeiliger Text mit Zeilenumbrüchen
- **Label:** Beschriftung oberhalb der Textarea (Pflicht)
- **Placeholder:** Optional - Beispieltext für erwarteten Inhalt
- **Helper Text:** Optional - Zusätzliche Hinweise oder Formatierungsregeln
- **Error Message:** Fehlermeldung bei ungültiger Eingabe
- **Character Counter:** Optional - Zeigt verbleibende Zeichen bei Längenbegrenzung
- **Resize Handle:** Visuelle Affordanz zum vertikalen Vergrößern/Verkleinern
- **Required Indicator:** Rotes Sternchen (\*) bei Pflichtfeldern

## Verwendung

Textareas werden verwendet für:

- **Längere Texteingaben:** Kommentare, Feedback, Nachrichten, Bewertungen
- **Beschreibungen:** Produktbeschreibungen, Projektdetails, Notizen
- **Formulare:** Kontaktformulare ("Ihre Nachricht"), Support-Tickets
- **Freitext-Antworten:** Offene Fragen in Umfragen

**Wann nicht verwenden:**

- Einzeilige kurze Eingaben (Name, E-Mail) → Text Input verwenden
- Strukturierte Daten → Spezifische Input-Typen verwenden
- Formatierter Text (fett, kursiv) → Rich Text Editor verwenden (separate Phase)

## Unterschied zu Text Input

Aspekt	Text Input	Textarea
Zeilen	Einzeilig	Mehrzeilig
Höhe	Fix (32/40/48px)	Variabel (min. 120px)
Zeilenumbrüche	Keine	Enter-Taste erzeugt neue Zeile
Resize	Nicht möglich	Vertikal möglich
Verwendung	Kurze strukturierte Daten	Längere Freitexte
Scroll	Horizontal bei Überlänge	Vertikal bei Überlänge

## Größen

Textareas verwenden nur **Medium** und **Large** Größen. Small ist nicht empfohlen, da Textareas für längere Inhalte gedacht sind.

### Medium (Empfohlen)

#### Spezifikationen:

- Min-Höhe: 120px (ca. 5 Zeilen bei 16px Schriftgröße)
- Padding: 12px horizontal, 8px vertikal
- Schriftgröße: 16px (fontSize.base)
- Zeilenhöhe: 1.5 (24px - lineHeight.normal)

#### Verwendung:

- Standard für die meisten Formulare
- Kommentarfelder, Feedback-Formulare
- Responsive Layouts (Desktop + Mobile)

#### Berechnung der Höhe:

- Schriftgröße: 16px
- Zeilenhöhe: 16px × 1.5 = 24px
- 5 Zeilen: 24px × 5 = 120px
- Plus Padding: 120px + (8px × 2) = 136px Gesamthöhe

### Large

#### Spezifikationen:

- Min-Höhe: 200px (ca. 8 Zeilen bei 18px Schriftgröße)
- Padding: 16px horizontal, 12px vertikal
- Schriftgröße: 18px (fontSize.lg)
- Zeilenhöhe: 1.5 (27px)

#### Verwendung:

- Mobile-optimierte Formulare
- Ausführliche Beschreibungen (Projektbriefings, Bewerbungen)
- Fokussierte Eingabeformulare mit wenigen Feldern

#### Vorteile:

- Großzügiger Eingabebereich reduziert Scrollen
- Bessere Lesbarkeit durch größere Schrift
- Signalisiert: "Hier wird ausführlicher Text erwartet"

## Zustände

Textareas verwenden dieselben Zustände wie Text Inputs für visuelle Konsistenz.

### Default

Visuelle Eigenschaften:

- Border: 1px solid neutral.300
- Background: neutral.white
- Text: neutral.900
- Placeholder: neutral.400
- Resize Handle: Rechts unten, neutral.400

Design Tokens:

```
input.field.background.default
input.field.border.default
input.field.text.default
textarea.minHeight
```

### Hover

Visuelle Eigenschaften:

- Border: 1px solid neutral.400
- Background: neutral.white
- Cursor: text

Design Tokens:

```
input.field.border.hover
```

### Focus

Visuelle Eigenschaften:

- Border: 2px solid hydrophon.blau.500
- Focus Ring: 3px hydrophon.blau.100
- Background: neutral.white

WCAG 2.4.7 Konformität:

- Deutlich sichtbarer Focus-Indikator
- 2px Border + 3px Ring = 5px Gesamtdicke
- :focus-visible für Tastatur-Navigation

Design Tokens:

```
input.field.border.focus
input.field.borderWidth.focus
input.focus.ring.color
input.focus.ring.width
```

## Error

### Visuelle Eigenschaften:

- Border: 1px solid color.error
- Background: neutral.white
- Error Message: Unterhalb, 12px, rot
- Optional: Error Icon (AlertCircle) rechts oben am Rand

### Verwendung:

- Pflichtfeld nicht ausgefüllt
- Minimale Zeichenanzahl nicht erreicht
- Maximale Zeichenanzahl überschritten
- Verbotene Inhalte (z.B. HTML-Tags, wenn nicht erlaubt)

### Design Tokens:

```
input.field.border.error
input.error.color
input.error.fontSize
```

## Success

### Visuelle Eigenschaften:

- Border: 1px solid color.success
- Background: neutral.white
- Optional: Success Icon (CheckCircle) rechts oben

### Verwendung:

- Seltener als bei Text Inputs
- Nur bei kritischen Validierungen (z.B. "Nachricht wurde gespeichert")

### Design Tokens:

```
input.field.border.success
input.success.color
```

## Disabled

### Visuelle Eigenschaften:

- Background: neutral.50
- Border: 1px solid neutral.200
- Text: neutral.600
- Opacity: 0.7
- Resize Handle: Ausgeblendet
- Cursor: not-allowed

**Design Tokens:**

```
input.field.background.disabled
input.field.border.disabled
input.field.text.disabled
```

## Höhenverhalten

Das Resize-Verhalten ist entscheidend für gute Textarea-UX.

### Resize-Optionen

**Empfohlen:** `resize: vertical`

- Benutzer kann Textarea vertikal vergrößern
- Verhindert horizontales Resize (bricht Layout)
- CSS: `resize: vertical;`
- Design Token: `input.textarea.resize`

**Alternativen:**

- `resize: none` - Keine Größenänderung möglich (nicht empfohlen, schränkt Benutzer ein)
- `resize: both` - Horizontal + vertikal (nicht empfohlen, bricht responsive Layouts)
- `resize: auto` - Browser-Default (meist beide Richtungen)

### Min-Height und Max-Height

**Min-Height:**

- Standard: 120px (ca. 5 Zeilen)
- Verhindert zu kleine Textareas
- CSS: `min-height: var(--input-textarea-min-height);`

**Max-Height (optional):**

- Verhindert übermäßig große Textareas
- Empfehlung: 400px (ca. 16 Zeilen bei 16px Schrift)
- Nach Überschreitung: Vertikales Scrollen
- CSS: `max-height: 400px; overflow-y: auto;`

### Auto-Growing Textareas (Optional)

Fortgeschrittene Pattern: Textarea wächst automatisch mit Inhalt.

**Vorteile:**

- Kein manuelles Resizing nötig
- Immer genau so groß wie Inhalt
- Reduziert Scrollen innerhalb Textarea

**Nachteile:**

- Komplexere JavaScript-Implementierung
- Layout-Shifts während Eingabe
- Potenziell sehr lange Seiten

**Implementierung:**

```
textarea.addEventListener('input', function() {
  this.style.height = 'auto';
  this.style.height = this.scrollHeight + 'px';
});
```

**Empfehlung:** Für Phase 3 bei statischen `resize: vertical` bleiben. Auto-Growing in separater Phase.

## Zeichenzähler

Bei Textareas mit Längenbegrenzung ist ein Zeichenzähler essenziell.

### Wann verwenden

- **Immer bei maxlength-Attribut:** Zeigt verbleibende Zeichen
- **Twitter-Style (280 Zeichen):** Kurze Nachrichten, Kommentare
- **Beschreibungen mit Limit:** Produktbeschreibungen (max. 500 Zeichen)
- **API-Limits:** Wenn Backend Längenbegrenzung hat

### Position und Darstellung

**Position:** Rechts unten, oberhalb Resize Handle

**Darstellung:**

Noch 240 Zeichen verfügbar

**Dynamisches Verhalten:**

- Über 20% verbleibend: `neutral.600` (grau)
- 10-20% verbleibend: `color.warning` (orange)
- Unter 10% verbleibend: `color.error` (rot)
- 0 Zeichen verbleibend: `color.error`, fett

**HTML-Struktur:**

```
<div class="textarea-wrapper">
  <textarea maxlength="280" id="comment"></textarea>
  <span class="character-counter" aria-live="polite">
    Noch 280 Zeichen verfügbar
  </span>
</div>
```

**Accessibility:**

- `aria-live="polite"` für Screen Reader Updates
- Text muss auch visuell sichtbar sein (nicht nur via ARIA)

## Accessibility (WCAG 2.1 AA)

Textareas folgen denselben Accessibility-Richtlinien wie Text Inputs.

## WCAG Anforderungen

### 3.3.2 Labels oder Anweisungen (Level A):

- Sichtbares Label oberhalb der Textarea
- Label mit `for` / `id` verknüpft
- Placeholder ist kein Ersatz für Label

### 3.3.3 Fehlervorschläge (Level AA):

- Error Messages mit konkreten Lösungen
- Beispiel: "Die Beschreibung muss mindestens 50 Zeichen lang sein (aktuell: 23)"

### 4.1.2 Name, Rolle, Wert (Level A):

- `role="textbox"` (implizit durch `<textarea>`)
- `aria-multiline="true"` (implizit)
- `aria-required="true"` bei Pflichtfeldern
- `aria-invalid="true"` bei Fehlern
- `aria-describedby` für Helper Text / Error Messages

## HTML-Struktur mit ARIA

Standard Textarea:

```
<div class="form-field">
  <label for="description" class="input-label">
    Beschreibung <span class="required">*</span>
  </label>
  <textarea
    id="description"
    name="description"
    class="textarea"
    rows="5"
    required
    aria-required="true"
    placeholder="Bitte beschreibe dein Anliegen ausführlich..."></textarea>
</div>
```

Mit Helper Text:

```
<div class="form-field">
  <label for="feedback">Feedback</label>
  <textarea
    id="feedback"
    aria-describedby="feedback-help"
    rows="5"></textarea>
  <span id="feedback-help" class="helper-text">
    Teile uns mit, was wir verbessern können (optional)
  </span>
</div>
```

Mit Zeichenzähler:

```
<div class="form-field">
  <label for="comment">Kommentar <span class="required">*</span></label>
  <div class="textarea-wrapper">
    <textarea
      id="comment"
      maxlength="280"
      aria-required="true"
      aria-describedby="comment-counter"
      rows="4"
    ></textarea>
    <span id="comment-counter" class="character-counter" aria-live="polite">
      Noch 280 Zeichen verfügbar
    </span>
  </div>
</div>
```

Error State:

```
<div class="form-field">
  <label for="message">Nachricht <span class="required">*</span></label>
  <textarea
    id="message"
    aria-invalid="true"
    aria-describedby="message-error"
    rows="5"
  ></textarea>
  <span id="message-error" class="error-text" role="alert">
    Die Nachricht muss mindestens 20 Zeichen lang sein (aktuell: 8)
  </span>
</div>
```

## Tastatur-Navigation

- **Tab:** Fokus in Textarea
- **Enter:** Neue Zeile (nicht Submit!)
- **Shift + Tab:** Fokus zum vorherigen Element
- **Ctrl/Cmd + Enter:** Optional - Formular absenden (bei Chat-UIs)

## Code-Beispiel

### CSS mit Design Tokens

```

/* Textarea Base Styles */
.textarea {
  /* Sizing */
  min-height: var(--input-textarea-min-height); /* 120px */
  width: 100%;
  padding: 8px var(--input-padding-x-md); /* 8px 12px */

  /* Typography */
  font-family: var(--input-base-font-family);
  font-size: var(--input-font-size-md); /* 16px */
  line-height: 1.5; /* 24px */
  color: var(--input-field-text-default);

  /* Visual */
  background-color: var(--input-field-background-default);
  border: var(--input-field-border-width-default) solid var(--input-field-border-default);
  border-radius: var(--input-base-border-radius); /* 4px */

  /* Interaction */
  resize: var(--input-textarea-resize); /* vertical */
  transition: var(--input-base-transition);
}

/* Placeholder */
.textarea::placeholder {
  color: var(--input-field-text-placeholder);
  opacity: 1;
}

/* Hover State */
.textarea:hover:not(:disabled) {
  border-color: var(--input-field-border-hover);
}

/* Focus State */
.textarea:focus-visible {
  outline: var(--input-field-border-width-focus) solid var(--input-field-border-focus);
  outline-offset: var(--input-focus-outline-offset);
  box-shadow: 0 0 0 var(--input-focus-ring-width) var(--input-focus-ring-color);
  border-color: var(--input-field-border-focus);
}

/* Error State */
.textarea[aria-invalid="true"] {
  border-color: var(--input-field-border-error);
}

/* Disabled State */
.textarea:disabled {
  background-color: var(--input-field-background-disabled);
  border-color: var(--input-field-border-disabled);
  color: var(--input-field-text-disabled);
}

```

```
  resize: none;
  cursor: not-allowed;
  opacity: 0.7;
}

/* Large Size Variant */
.textarea--large {
  min-height: 200px;
  padding: 12px var(--input-padding-x-lg); /* 12px 16px */
  font-size: var(--input-font-size-lg); /* 18px */
}

/* Character Counter */
.character-counter {
  display: block;
  text-align: right;
  font-size: var(--input-helper-font-size); /* 12px */
  color: var(--input-helper-color);
  margin-top: var(--input-helper-margin-top); /* 4px */
}

.character-counter--warning {
  color: var(--color-warning);
}

.character-counter--error {
  color: var(--input-error-color);
  font-weight: 500;
}
```

## JavaScript für Zeichenzähler

```

function initCharacterCounter(textareaId) {
  const textarea = document.getElementById(textareaId);
  const counter = textarea.nextElementSibling; // Assumes counter is next sibling
  const maxLength = textarea.getAttribute('maxlength');

  if (!maxLength || !counter) return;

  function updateCounter() {
    const remaining = maxLength - textarea.value.length;
    const percentage = (remaining / maxLength) * 100;

    counter.textContent = `Noch ${remaining} Zeichen verfügbar`;

    // Color coding based on remaining characters
    counter.classList.remove('character-counter--warning', 'character-counter--error');

    if (percentage <= 10) {
      counter.classList.add('character-counter--error');
    } else if (percentage <= 20) {
      counter.classList.add('character-counter--warning');
    }
  }

  textarea.addEventListener('input', updateCounter);
  updateCounter(); // Initial state
}

```

## Design Tokens Referenz

Token Name	Wert	Verwendung
input.textarea.minHeight	120px	Minimale Textarea-Höhe (ca. 5 Zeilen)
input.textarea.resize	vertical	Resize-Verhalten (nur vertikal)
input.fontSize.md	16px	Schriftgröße Medium
input.fontSize.lg	18px	Schriftgröße Large
input.padding.x.md	12px	Horizontales Padding Medium
input.padding.x.lg	16px	Horizontales Padding Large
input.field.background.default	neutral.white	Hintergrund Standard
input.field.border.default	neutral.300	Border Standard
input.field.border.hover	neutral.400	Border Hover
input.field.border.focus	hydrophon.blau.500	Border Focus
input.field.border.error	color.error	Border Error
input.field.text.default	neutral.900	Text-Farbe
input.field.text.placeholder	neutral.400	Placeholder-Farbe
input.base.borderRadius	4px	Eckenradius

**Version:** 1.0 **Letzte Aktualisierung:** 2026-01-29 **Phase:** 03 - Forms & Data Input **Komponente:** FORM-02 - Textarea

---

# Select / Dropdown

---

Auswahlfeld für die Selektion einer einzelnen Option aus einer vordefinierten Liste von Möglichkeiten.

## Übersicht

Select-Komponenten (auch Dropdown genannt) ermöglichen Benutzern die Auswahl einer einzelnen Option aus einer Liste. Sie sind platzsparend, da nur die ausgewählte Option sichtbar ist, bis der Benutzer die Liste öffnet.

### Native vs. Custom Select

**Empfehlung:** Native `<select>` bevorzugen

Wann immer möglich, sollte das native HTML `<select>`-Element verwendet werden:

**Vorteile von Native Select:**

- **✓ Accessibility:** Vollständig barrierefrei ohne zusätzlichen Code
- **✓ Tastatur-Navigation:** Browser-native Bedienung (Pfeiltasten, Enter, Escape)
- **✓ Screen Reader:** Perfekte Unterstützung ohne ARIA
- **✓ Mobile:** Nutzt native OS-Picker (iOS Wheel, Android Dropdown)
- **✓ Performance:** Keine JavaScript-Overhead
- **✓ Zuverlässigkeit:** Kein Risiko für Custom-Bugs

**Nachteile von Native Select:**

- **✗ Styling-Grenzen:** Limitierte visuelle Anpassbarkeit (insbesondere Options)
- **✗ Keine Icons in Options:** Nur Text möglich
- **✗ Keine Gruppierungs-Visualisierung:** `<optgroup>` hat Standard-Styling
- **✗ Kein Multi-Select UX:** `multiple`-Attribut hat schlechte UX

**Wann Custom Select verwenden:**

- Icons oder Bilder in Options notwendig
- Komplexe Option-Layouts (z.B. Name + Beschreibung)
- Multi-Select mit Checkboxen
- Autocomplete/Search innerhalb Dropdown
- Sehr spezifisches Brand-Styling erforderlich

## Anatomie

Ein vollständiges Native Select besteht aus:

- **Container:** Border + Background (konsistent mit Text Input)
- **Selected Value:** Aktuell ausgewählter Text
- **Dropdown Icon:** ChevronDown Icon (20px) rechts
- **Label:** Beschriftung oberhalb (Pflicht)
- **Options List:** Dropdown-Liste bei Klick (Browser-styled)
- **Helper Text:** Optional - Zusätzliche Hinweise
- **Error Message:** Fehlermeldung bei ungültiger Auswahl

- **Required Indicator:** Rotes Sternchen (\*) bei Pflichtfeldern

## Verwendung

Select-Komponenten werden verwendet für:

- **Vordefinierte Optionen:** Land, Stadt, Kategorie, Sortierung
- **Status-Auswahl:** Bestellstatus, Priorität, Phase
- **Listen mit 3-15 Optionen:** Nicht zu wenig (dann Radio), nicht zu viel (dann Autocomplete)
- **Mobile-freundliche Auswahl:** Native Picker besser als Custom-Dropdowns

**Wann nicht verwenden:**

- 2 Optionen → Radio Buttons verwenden (direktere Auswahl)
- Über 15 Optionen → Autocomplete/Search Input verwenden
- Multi-Select → Custom Multi-Select mit Checkboxen (separate Komponente)
- Ja/Nein Entscheidung → Toggle Switch oder Checkbox verwenden
- Freitext-Eingabe → Text Input verwenden

## Größen

Select-Komponenten verwenden dieselben Größen wie Text Inputs für visuelle Konsistenz.

### Small (32px)

**Spezifikationen:**

- Höhe: 32px (input.height.sm)
- Padding horizontal: 8px links, 32px rechts (Platz für Icon)
- Schriftgröße: 14px (input.font.size.sm)
- Icon-Größe: 20px (input.select.iconSize)

**Verwendung:**

- Kompakte Desktop-Interfaces
- Tabellen-Filter, Sortierungs-Dropdowns
- Admin-Panels mit hoher Informationsdichte

**Wichtig:** Nur Desktop - Touch-Target unter 44px nicht mobile-friendly.

### Medium (40px) - Standard

**Spezifikationen:**

- Höhe: 40px (input.height.md)
- Padding horizontal: 12px links, 36px rechts
- Schriftgröße: 16px (input.font.size.md)
- Icon-Größe: 20px

**Verwendung:**

- Standard für Formulare
- Responsive Layouts (Desktop + Mobile)
- Checkout-Prozesse, Registrierung

**Vorteile:**

- Gute Balance zwischen Kompaktheit und Usability
- 40px Höhe bietet ausreichende Touch-Ergonomie

## Large (48px)

### Spezifikationen:

- Höhe: 48px (input.height.lg)
- Padding horizontal: 16px links, 40px rechts
- Schriftgröße: 18px (input.fontSize.lg)
- Icon-Größe: 20px

### Verwendung:

- Mobile-optimierte Formulare
- Hero-Sections mit prominenten Selects (z.B. Produktfilter)
- CTAs und wichtige Auswahlfelder

### Vorteile:

- Übertrifft WCAG AAA Touch-Target (44px)
- Großzügige Klickfläche
- Gute Lesbarkeit durch 18px Schrift

## Zustände

Select-Komponenten folgen denselben Zuständen wie Text Inputs.

### Default

Der Standard-Zustand eines Selects ohne Benutzerinteraktion.

#### Visuelle Eigenschaften:

- Border: 1px solid neutral.300
- Background: neutral.white
- Text: neutral.900 (ausgewählter Wert)
- Icon: neutral.600 , ChevronDown, 20px, rechts positioniert
- Cursor: pointer

#### Design Tokens:

```
input.field.background.default
input.field.border.default
input.field.text.default
input.select.iconColor.default
input.select.iconSize
```

**Wichtig:** Placeholder-Option für "Bitte wählen" verwenden:

```
<option value="" disabled selected>Bitte wählen...</option>
```

### Hover

Zustand wenn der Mauszeiger über dem Select schwebt.

#### Visuelle Eigenschaften:

- Border: 1px solid neutral.400

- Background: neutral.white
- Icon: Unverändert neutral.600
- Cursor: pointer

#### Design Tokens:

```
input.field.border.hover
```

### Focus / Open

Zustand wenn Select fokussiert ist oder Dropdown-Liste geöffnet wurde.

#### Visuelle Eigenschaften:

- Border: 2px solid hydrophon.blau.500
- Focus Ring: 3px hydrophon.blau.100
- Background: neutral.white
- Icon: Optional rotiert um 180° (zeigt nach oben bei geöffnetem Dropdown)

#### Design Tokens:

```
input.field.border.focus
input.field.borderWidth.focus
input.focus.ring.color
input.focus.ring.width
```

#### Icon-Rotation (optional):

```
select:focus + .select-icon {
  transform: rotate(180deg);
}
```

#### WCAG 2.4.7 Konformität:

- Deutlich sichtbarer Focus-Indikator
- :focus-visible für Tastatur-Navigation

### Error

Zustand bei ungültiger oder fehlender Auswahl (z.B. Pflichtfeld nicht ausgefüllt).

#### Visuelle Eigenschaften:

- Border: 1px solid color.error
- Background: neutral.white
- Error Icon: Optional - AlertCircle links neben Dropdown-Icon
- Error Message: Unterhalb, 12px, rot

#### Design Tokens:

```
input.field.border.error
input.error.color
input.error.fontSize
```

**Verwendung:**

- Pflichtfeld nicht ausgefüllt (Value = "")
- Ungültige Auswahl nach Business-Logik (z.B. nicht verfügbare Option)

**Fehlermeldungen:**

- ✗ Schlecht: "Ungültig"
- ✓ Gut: "Bitte wähle ein Land aus"
- ✓ Gut: "Diese Option ist aktuell nicht verfügbar"

**Disabled**

Zustand für nicht-auswählbare Selects (z.B. abhängige Felder).

**Visuelle Eigenschaften:**

- Background: neutral.50
- Border: 1px solid neutral.200
- Text: neutral.600
- Icon: neutral.400 (gedämpft)
- Opacity: 0.7
- Cursor: not-allowed

**Design Tokens:**

```
input.field.background.disabled
input.field.border.disabled
input.field.text.disabled
input.select.iconColor.disabled
```

**Accessibility:**

- `disabled` Attribut setzen (verhindert Fokus und Auswahl)
- Optional: Tooltip mit Erklärung warum deaktiviert

**Dropdown Icon**

Das Dropdown-Icon ist essenziell für die Erkennbarkeit eines Select-Elements.

**Icon-Spezifikation**

Icon: Lucide ChevronDown

- Größe: 20px (`input.select.iconSize`)
- Stroke Width: 2px (Lucide Standard)
- Farbe: `currentColor` oder `neutral.600`
- Position: Rechts, vertikal zentriert, 8-12px vom rechten Rand

**Design Tokens:**

```
input.select.iconSize: 20px
input.select.iconColor.default: neutral.600
input.select.iconColor.disabled: neutral.400
```

## Icon-Position

Absolute Positionierung:

```
.select-wrapper {
  position: relative;
}

.select-icon {
  position: absolute;
  right: 12px;
  top: 50%;
  transform: translateY(-50%);
  pointer-events: none; /* Icon nicht klickbar, nur Select */
  color: var(--input-select-icon-color-default);
}
```

Native Select Icon ausblenden:

```
select {
  appearance: none; /* Entfernt Browser-Default-Icon */
  -webkit-appearance: none;
  -moz-appearance: none;
}
```

## Options Styling

Wichtige Einschränkung: Native `<option>`-Elemente haben sehr limitierte Styling-Möglichkeiten.

### Was funktioniert

- **Text:** Schriftart, Größe (erbt von `<select>`)
- **Farbe:** `color` und `background-color` (sehr limitiert, Browser-abhängig)
- **Disabled Options:** `<option disabled>` hat gedämpftes Grau

### Was nicht funktioniert

- ❌ Icons in Options
- ❌ Komplexe Layouts (z.B. Name + Beschreibung zweizeilig)
- ❌ Custom Checkboxen (bei Multi-Select)
- ❌ Borders, Shadows, Hover-States

## Optgroup (Gruppen)

Native Unterstützung für Gruppierung:

```
<select>
  <optgroup label="Europa">
    <option value="de">Deutschland</option>
    <option value="fr">Frankreich</option>
  </optgroup>
  <optgroup label="Nordamerika">
    <option value="us">USA</option>
    <option value="ca">Kanada</option>
  </optgroup>
</select>
```

**Styling:** Browser-Default (grau, eingerückt) - schwer anzupassen.

## Custom Select (Wann und Wie)

Wenn native Selects nicht ausreichen, muss eine Custom-Implementierung erfolgen.

### Anforderungen an Custom Select

Pflicht-Features:

- Vollständige Tastatur-Navigation (Pfeiltasten, Enter, Escape, Home, End)
- ARIA-Attribute ( `role="listbox"` , `aria-expanded` , `aria-activeitem` )
- Screen Reader Support (alle Options vorgelesen)
- Focus Management (Focus bleibt beim Öffnen/Schließen erhalten)
- Click-Outside zum Schließen
- Escape-Taste zum Schließen

Empfohlene Libraries:

- Headless UI (React/Vue): Accessibility eingebaut, unstyled
- Radix UI (React): Composable Select Primitive
- Downshift (React): Flexible Select/Autocomplete
- Tom Select: Vanilla JS, sehr feature-reich

### Anti-Pattern: Custom Div-Dropdowns

Niemals Custom Dropdowns ohne ARIA bauen!

```
<!-- ❌ FALSCH - Nicht barrierefrei -->
<div class="select" onclick="toggleDropdown()">
  <span>Bitte wählen</span>
  <div class="dropdown" id="dropdown">
    <div onclick="selectOption('option1')">Option 1</div>
    <div onclick="selectOption('option2')">Option 2</div>
  </div>
</div>
```

Probleme:

- Keine Tastatur-Navigation
- Screen Reader erkennt keine Options

- Kein Focus Management
- Mobile: Kein native Picker
- Performance: Event-Listener auf jedem Item

Wenn Custom, dann richtig:

```
<!-- ✓ RICHTIG - Mit ARIA -->
<div class="select-wrapper">
  <button
    type="button"
    role="combobox"
    aria-expanded="false"
    aria-controls="listbox"
    aria-haspopup="listbox"
    id="select-button"
  >
    <span>Bitte wählen</span>
    <svg aria-hidden="true"><!-- ChevronDown Icon --></svg>
  </button>

  <ul
    role="listbox"
    id="listbox"
    aria-labelledby="select-button"
    hidden
  >
    <li role="option" aria-selected="false" id="option-1">Option 1</li>
    <li role="option" aria-selected="false" id="option-2">Option 2</li>
  </ul>
</div>
```

## Accessibility (WCAG 2.1 AA)

Native Selects sind von Haus aus barrierefrei. Custom Selects benötigen umfangreiche ARIA-Implementation.

### WCAG Anforderungen

#### 3.3.2 Labels oder Anweisungen (Level A):

- Sichtbares Label oberhalb
- Label mit `for` / `id` verknüpft

#### 4.1.2 Name, Rolle, Wert (Level A):

- Native Select: Automatisch erfüllt
- Custom Select: `role="listbox"`, `role="option"`, `aria-selected` erforderlich

#### 2.1.1 Tastatur (Level A):

- Alle Funktionen müssen per Tastatur bedienbar sein
- Native Select: Browser-native Unterstützung
- Custom Select: Pfeiltasten, Enter, Escape, Tab implementieren

## HTML-Struktur Native Select

Standard Select:

```
<div class="form-field">
  <label for="country" class="input-label">
    Land <span class="required">*</span>
  </label>
  <div class="select-wrapper">
    <select id="country" name="country" required aria-required="true">
      <option value="" disabled selected>Bitte wählen...</option>
      <option value="de">Deutschland</option>
      <option value="at">Österreich</option>
      <option value="ch">Schweiz</option>
    </select>
    <svg class="select-icon" width="20" height="20">
      <!-- ChevronDown Icon -->
    </svg>
  </div>
</div>
```

Mit Helper Text:

```
<div class="form-field">
  <label for="priority">Priorität</label>
  <div class="select-wrapper">
    <select id="priority" aria-describedby="priority-help">
      <option value="low">Niedrig</option>
      <option value="medium">Mittel</option>
      <option value="high">Hoch</option>
    </select>
    <svg class="select-icon"><!-- Icon --></svg>
  </div>
  <span id="priority-help" class="helper-text">
    Bestimmt die Bearbeitungsreihenfolge
  </span>
</div>
```

Error State:

```
<div class="form-field">
  <label for="category">Kategorie <span class="required">*</span></label>
  <div class="select-wrapper">
    <select
      id="category"
      aria-invalid="true"
      aria-describedby="category-error"
    >
      <option value="" disabled selected>Bitte wählen...</option>
      <option value="cat1">Kategorie 1</option>
    </select>
    <svg class="select-icon"><!-- Icon --></svg>
  </div>
  <span id="category-error" class="error-text" role="alert">
    Bitte wähle eine Kategorie aus
  </span>
</div>
```

### Tastatur-Navigation (Native Select)

- **Tab:** Fokus auf Select
- **Space / Enter:** Dropdown öffnen
- **Pfeil Runter / Pfeil Hoch:** Nächste/Vorherige Option
- **Home:** Erste Option
- **End:** Letzte Option
- **Buchstabe tippen:** Spring zu Option die mit Buchstabe beginnt
- **Escape:** Dropdown schließen (ohne Auswahl)
- **Enter (bei geöffneter Liste):** Auswahl bestätigen und schließen

## Code-Beispiel

### CSS mit Design Tokens

```

/* Select Wrapper */
.select-wrapper {
  position: relative;
  display: inline-block;
  width: 100%;
}

/* Native Select */
select {
  /* Remove default styling */
  appearance: none;
  -webkit-appearance: none;
  -moz-appearance: none;

  /* Sizing */
  height: var(--input-height-md); /* 40px */
  width: 100%;
  padding: 0 36px 0 var(--input-padding-x-md); /* Right padding for icon */

  /* Typography */
  font-family: var(--input-base-font-family);
  font-size: var(--input-font-size-md); /* 16px */
  color: var(--input-field-text-default);

  /* Visual */
  background-color: var(--input-field-background-default);
  border: var(--input-field-border-width-default) solid var(--input-field-border-default);
  border-radius: var(--input-base-border-radius); /* 4px */

  /* Interaction */
  cursor: pointer;
  transition: var(--input-base-transition);
}

/* Dropdown Icon */
.select-icon {
  position: absolute;
  right: 12px;
  top: 50%;
  transform: translateY(-50%);
  width: var(--input-select-icon-size); /* 20px */
  height: var(--input-select-icon-size);
  color: var(--input-select-icon-color-default);
  pointer-events: none; /* Click goes through to select */
  transition: transform 150ms ease-in-out;
}

/* Hover State */
select:hover:not(:disabled) {
  border-color: var(--input-field-border-hover);
}

```

```

/* Focus State */
select:focus-visible {
  outline: var(--input-field-border-width-focus) solid var(--input-field-border-focus);
  outline-offset: var(--input-focus-outline-offset);
  box-shadow: 0 0 0 var(--input-focus-ring-width) var(--input-focus-ring-color);
  border-color: var(--input-field-border-focus);
}

/* Icon rotation when open (requires JavaScript to add class) */
select:focus + .select-icon {
  transform: translateY(-50%) rotate(180deg);
}

/* Error State */
select[aria-invalid="true"] {
  border-color: var(--input-field-border-error);
}

/* Disabled State */
select:disabled {
  background-color: var(--input-field-background-disabled);
  border-color: var(--input-field-border-disabled);
  color: var(--input-field-text-disabled);
  cursor: not-allowed;
  opacity: 0.7;
}

select:disabled + .select-icon {
  color: var(--input-select-icon-color-disabled);
}

/* Size Variants */
select.select--small {
  height: var(--input-height-sm); /* 32px */
  padding-left: var(--input-padding-x-sm); /* 8px */
  padding-right: 32px;
  font-size: var(--input-font-size-sm); /* 14px */
}

select.select--large {
  height: var(--input-height-lg); /* 48px */
  padding-left: var(--input-padding-x-lg); /* 16px */
  padding-right: 40px;
  font-size: var(--input-font-size-lg); /* 18px */
}

/* Option Styling (very limited) */
option {
  padding: 8px;
}

option:disabled {
  color: var(--input-field-text-disabled);
}

/* Optgroup Styling (even more limited) */
optgroup {
  font-weight: var(--input-label-font-weight); /* 500 */
}

```

```
    color: var(--input-label-color-default);  
}
```

## JavaScript für Icon-Rotation

```
// Optional: Rotate icon when dropdown is open  
document.querySelectorAll('select').forEach(select => {  
  const icon = select.nextElementSibling; // Assumes icon is next sibling  
  
  select.addEventListener('focus', () => {  
    if (icon && icon.classList.contains('select-icon')) {  
      icon.style.transform = 'translateY(-50%) rotate(180deg)';  
    }  
  });  
  
  select.addEventListener('blur', () => {  
    if (icon && icon.classList.contains('select-icon')) {  
      icon.style.transform = 'translateY(-50%) rotate(0deg)';  
    }  
  });  
});
```

## Design Tokens Referenz

Token Name	Wert	Verwendung
input.height.sm	32px	Select-Höhe Small
input.height.md	40px	Select-Höhe Medium (Standard)
input.height.lg	48px	Select-Höhe Large
input.padding.x.sm	8px	Horizontales Padding Small
input.padding.x.md	12px	Horizontales Padding Medium
input.padding.x.lg	16px	Horizontales Padding Large
input.font.size.sm	14px	Schriftgröße Small
input.font.size.md	16px	Schriftgröße Medium
input.font.size.lg	18px	Schriftgröße Large
input.select.iconSize	20px	Dropdown Icon Größe
input.select.iconColor.default	neutral.600	Icon-Farbe Standard
input.select.iconColor.disabled	neutral.400	Icon-Farbe Disabled
input.field.background.default	neutral.white	Hintergrund Standard
input.field.border.default	neutral.300	Border Standard
input.field.border.hover	neutral.400	Border Hover
input.field.border.focus	hydrophon.blau.500	Border Focus
input.field.border.error	color.error	Border Error
input.field.border.disabled	neutral.200	Border Disabled
input.field.text.default	neutral.900	Text-Farbe
input.base.borderRadius	4px	Eckenradius

---

Version: 1.0 Letzte Aktualisierung: 2026-01-29 Phase: 03 - Forms & Data Input Komponente: FORM-03 - Select / Dropdown

---

# Checkbox

---

Ermöglicht Auswahl von einer oder mehreren Optionen aus einer Liste. Checkboxen erlauben unabhängige Entscheidungen - jede Option kann einzeln an- oder abgewählt werden.

## Übersicht

### Anatomie

Eine Checkbox besteht aus folgenden Elementen:

- **Native Input** - Natives `<input type="checkbox">` Element (visuell versteckt, Accessibility erhalten)
- **Custom Visual** - Gestylter Container, der das native Element überlagert
- **Check-Icon** - Lucide Check-Icon (14px) bei angekreuztem Zustand
- **Label-Text** - Beschreibender Text neben der Checkbox
- **Helper Text (optional)** - Zusätzliche Erklärung unter der Checkbox
- **Error Message (optional)** - Validierungsfehlermeldung bei Fehler

### Verwendung

Checkboxen eignen sich für:

- **Mehrfachauswahl in Formularen** - z.B. "Welche Newsletter möchten Sie erhalten?"
- **Einverständniserklärungen** - AGB akzeptieren, Datenschutz zustimmen
- **Filter und Einstellungen** - Produktfilter, Account-Einstellungen
- **Todo-Listen** - Aufgaben als erledigt markieren
- **Optionale Features aktivieren/deaktivieren**

Nicht verwenden für:

- Exklusive Auswahl (nur eine Option möglich) → verwende Radio Buttons
- Binäre Ein/Aus-Schalter mit sofortiger Wirkung → verwende Toggle Switch

## Größen

### Default (20px)

- Checkbox: 20x20px
- Check-Icon: 14px
- Verwendung: Standard für die meisten Formulare
- Desktop und Mobile geeignet

### Large (24px)

- Checkbox: 24x24px
- Check-Icon: 16px
- Verwendung: Mobile-optimierte Interfaces, verbesserte Touch-Targets

- Erfüllt WCAG AAA 44px Empfehlung in Kombination mit Label-Klickfläche

## Zustände

### Unchecked Default

- Background: neutral.white
- Border: neutral.300 (2px)
- Visuell: Leere weiße Box mit grauem Rahmen

### Unchecked Hover

- Background: neutral.white
- Border: neutral.400 (dunkler)
- Feedback: Subtile Verdunklung des Rahmens

### Checked

- Background: hydrophon.blau.500
- Border: hydrophon.blau.500
- Icon: Weißes Checkmark (Check aus Lucide Icon Set)
- Visuell: Blaue Box mit weißem Haken

### Checked Hover

- Background: hydrophon.blau.600 (dunkler)
- Border: hydrophon.blau.600
- Icon: Weißes Checkmark
- Feedback: Verdunklung bei Hover über angekreuzte Checkbox

### Focus

- Outline: 2px hydrophon.blau.300
- Offset: 2px
- Nur bei :focus-visible (Tastatur-Navigation)
- Kombiniert mit aktuellem Zustand (checked/unchecked)

**Wichtig:** Focus-Indikator erscheint NUR bei Tastatur-Navigation, nicht bei Maus-Klick. Dies verhindert visuelles Rauschen und erfüllt gleichzeitig WCAG 2.4.7 (Focus sichtbar).

### Error

- Border: color.error (rot)
- Error-Message unter der Checkbox oder Checkbox-Gruppe
- Kombiniert mit aktuellem Zustand (checked/unchecked)
- Bei Gruppen: Error-Message bezieht sich auf die gesamte Gruppe

### Disabled (Unchecked)

- Background: neutral.50 (leicht grau)
- Border: neutral.200 (gedämpft)

- Label: `neutral.400` (gedämpft)
- Cursor: `not-allowed`
- Nicht interaktiv

### Disabled (Checked)

- Background: `neutral.300` (gedämpftes Grau)
- Border: `neutral.300`
- Icon: Weißes Checkmark (reduzierter Kontrast)
- Label: `neutral.400`
- Zeigt vorherige Auswahl, aber nicht änderbar

## Custom-Styling-Pattern

### Accessibility-konformes Verstecken

Das native `<input type="checkbox">` Element MUSS im DOM verbleiben, um Screenreader-Kompatibilität und Tastatur-Navigation zu gewährleisten.

**RICHTIG:** `opacity: 0` behält Accessibility

```
.checkbox-native {
  opacity: 0;
  position: absolute;
  width: 1px;
  height: 1px;
  pointer-events: none;
}
```

Diese Technik:

- Entfernt visuell das native Element
- Behält es im Accessibility Tree (Screenreader erkennen es)
- Erhält Tastatur-Navigation (Space-Taste funktioniert)
- Erhält native Formular-Validation

**FALSCH:** `display: none` entfernt aus Accessibility Tree

```
.checkbox-native {
  display: none; /* NIEMALS! */
}
```

Diese Technik:

- Entfernt Element komplett aus DOM
- Screenreader können es nicht erkennen
- Tastatur-Navigation funktioniert nicht
- Formular-Validation bricht
- WCAG 4.1.2 Verletzung

## Focus-Weiterleitung

Der Focus-Zustand des nativen Inputs wird auf das Custom Visual weitergeleitet:

```
/* Focus auf nativem Input stylt Custom Visual */
.checkbox-native:focus-visible + .checkbox-custom {
  outline: 2px solid var(--checkbox-focus-outline-color);
  outline-offset: 2px;
}

/* Checked-State */
.checkbox-native:checked + .checkbox-custom {
  background-color: var(--checkbox-background-checked);
  border-color: var(--checkbox-border-checked);
}

/* Checked + Hover kombiniert */
.checkbox-native:checked + .checkbox-custom:hover {
  background-color: var(--checkbox-background-checked-hover);
}

/* Disabled State */
.checkbox-native:disabled + .checkbox-custom {
  background-color: var(--checkbox-background-disabled);
  border-color: var(--checkbox-border-disabled);
  cursor: not-allowed;
}

/* Disabled + Checked kombiniert */
.checkbox-native:disabled:checked + .checkbox-custom {
  background-color: var(--checkbox-background-checked-disabled);
}
```

## Check-Icon Anzeige

Das Check-Icon erscheint nur im checked-Zustand:

```
.checkbox-icon {
  opacity: 0;
  transition: opacity 150ms ease-in-out;
}

.checkbox-native:checked + .checkbox-custom .checkbox-icon {
  opacity: 1;
}
```

## Accessibility

### WCAG Anforderungen

#### 4.1.2 Name, Rolle, Wert

- Native `<input type="checkbox">` behält automatisch Checkbox-Rolle

- Screenreader kündigen korrekt an: "checkbox, not checked" / "checkbox, checked"
- Keine zusätzlichen ARIA-Attribute nötig

#### 2.4.7 Focus sichtbar

- `:focus-visible` mit 2px Outline
- 2px Offset für klare Trennung vom Element
- 3:1 Kontrast (hellblau auf weiß)

#### 1.4.3 Kontrast

- Text-Kontrast: 4.5:1 (Label auf weißem Hintergrund)
- UI-Komponenten-Kontrast: 3:1 (Border auf weißem Hintergrund)
- Checked-State: Blau mit weißem Icon für maximalen Kontrast

#### 3.3.2 Labels

- Sichtbares Label für jede Checkbox
- `<label>` Element mit `for` Attribut oder umschließend
- Label beschreibt klar, was angekreuzt wird

#### 2.5.5 Touch-Target-Größe (WCAG 2.2)

- Minimum: 44x44px (AAA-Empfehlung)
- Default Checkbox (20px) + Label-Klickfläche erfüllt dies
- Large Checkbox (24px) für reine Icon-Darstellung

### Native Vorteile

Durch die Verwendung des nativen `<input type="checkbox">` Elements erhält man automatisch:

- **Space-Taste** - Toggelt Checkbox (checked ↔ unchecked)
- **Tab-Navigation** - Fokussiert nächste/vorherige Checkbox
- **Screenreader-Ansage** - "checkbox, not checked" / "checkbox, checked"
- **Formular-Validation** - `required` Attribut funktioniert nativ
- **Formular-Submission** - Wert wird automatisch mitgesendet
- **Keine ARIA-Attribute nötig** - Native Semantik ist ausreichend

### HTML-Struktur

Empfohlene Struktur mit umschließendem Label:

```
<label class="checkbox-container">
  <input type="checkbox" class="checkbox-native" />
  <span class="checkbox-custom">
    <svg class="checkbox-icon"><!-- Check Icon --></svg>
  </span>
  <span class="checkbox-label">Option akzeptieren</span>
</label>
```

Alternative mit `for`-Attribut:

```
<div class="checkbox-wrapper">
  <input type="checkbox" id="terms" class="checkbox-native" />
  <label for="terms" class="checkbox-custom-container">
    <span class="checkbox-custom">
      <svg class="checkbox-icon"><!-- Check Icon --></svg>
    </span>
    <span class="checkbox-label">AGB akzeptieren</span>
  </label>
</div>
```

Mit Helper Text:

```
<label class="checkbox-container">
  <input type="checkbox" class="checkbox-native" aria-describedby="privacy-helper" />
  <span class="checkbox-custom">
    <svg class="checkbox-icon"><!-- Check Icon --></svg>
  </span>
  <span class="checkbox-label">Datenschutz akzeptieren</span>
</label>
<p id="privacy-helper" class="checkbox-helper">
  Wir verwenden Ihre Daten nur zur Bearbeitung Ihrer Anfrage
</p>
```

Mit Error-Message:

```
<label class="checkbox-container checkbox-error">
  <input type="checkbox" class="checkbox-native" aria-describedby="terms-error" aria-invalid="true" />
  <span class="checkbox-custom">
    <svg class="checkbox-icon"><!-- Check Icon --></svg>
  </span>
  <span class="checkbox-label">AGB akzeptieren</span>
</label>
<p id="terms-error" class="checkbox-error-message">
  Bitte akzeptieren Sie die AGB, um fortzufahren
</p>
```

## Checkbox-Gruppe

### Mit Fieldset/Legend

Für mehrere zusammengehörige Checkboxen verwende `<fieldset>` und `<legend>`:

```

<fieldset>
  <legend>Interessen auswählen</legend>

  <label class="checkbox-container">
    <input type="checkbox" name="interests" value="produktnews" />
    <span class="checkbox-custom">
      <svg class="checkbox-icon"><!-- Check Icon --></svg>
    </span>
    <span class="checkbox-label">Produktneuheiten</span>
  </label>

  <label class="checkbox-container">
    <input type="checkbox" name="interests" value="events" />
    <span class="checkbox-custom">
      <svg class="checkbox-icon"><!-- Check Icon --></svg>
    </span>
    <span class="checkbox-label">Veranstaltungen</span>
  </label>

  <label class="checkbox-container">
    <input type="checkbox" name="interests" value="angebote" />
    <span class="checkbox-custom">
      <svg class="checkbox-icon"><!-- Check Icon --></svg>
    </span>
    <span class="checkbox-label">Sonderangebote</span>
  </label>
</fieldset>

```

## Fehler bei Gruppen

Bei Checkbox-Gruppen erscheint die Error-Message unter der gesamten Gruppe:

```

<fieldset aria-describedby="interests-error">
  <legend>Mindestens eine Option auswählen *</legend>

  <!-- Checkboxen wie oben -->

  <p id="interests-error" class="checkbox-error-message">
    Bitte wählen Sie mindestens eine Interessenskategorie aus
  </p>
</fieldset>

```

**Accessibility-Hinweis:** `aria-describedby` auf `<fieldset>` verknüpft Error-Message mit der gesamten Gruppe.

## Code-Beispiel

Vollständiges HTML + CSS Beispiel:

```
<!DOCTYPE html>
<html lang="de">
<head>
<style>
.checkbox-container {
  display: inline-flex;
  align-items: flex-start;
  gap: var(--checkbox-label-gap);
  cursor: pointer;
  position: relative;
}

.checkbox-native {
  opacity: 0;
  position: absolute;
  width: 1px;
  height: 1px;
  pointer-events: none;
}

.checkbox-custom {
  display: flex;
  align-items: center;
  justify-content: center;
  width: var(--checkbox-size-default);
  height: var(--checkbox-size-default);
  border: var(--checkbox-border-width) solid var(--checkbox-border-default);
  border-radius: var(--checkbox-border-radius);
  background-color: var(--checkbox-background-default);
  transition: all 150ms ease-in-out;
  flex-shrink: 0;
}

.checkbox-icon {
  width: var(--checkbox-icon-size);
  height: var(--checkbox-icon-size);
  color: var(--checkbox-icon-color);
  opacity: 0;
  transition: opacity 150ms ease-in-out;
}

/* Hover */
.checkbox-custom:hover {
  border-color: var(--checkbox-border-hover);
}

/* Checked */
.checkbox-native:checked + .checkbox-custom {
  background-color: var(--checkbox-background-checked);
  border-color: var(--checkbox-border-checked);
}

.checkbox-native:checked + .checkbox-custom .checkbox-icon {
  opacity: 1;
}

/* Checked + Hover */

```

```

.checkbox-native:checked + .checkbox-custom {
  background-color: var(--checkbox-background-checked);
}

/* Focus */
.checkbox-native:focus-visible + .checkbox-custom {
  outline: var(--checkbox-focus-outline-width) solid var(--checkbox-focus-outline-color);
  outline-offset: var(--checkbox-focus-outline-offset);
}

/* Disabled */
.checkbox-native:disabled + .checkbox-custom {
  background-color: var(--checkbox-background-disabled);
  border-color: var(--checkbox-border-disabled);
  cursor: not-allowed;
}

.checkbox-native:disabled:checked + .checkbox-custom {
  background-color: var(--checkbox-background-checked-disabled);
}

.checkbox-native:disabled ~ .checkbox-label {
  color: var(--checkbox-label-color-disabled);
}

/* Error */
.checkbox-container.checkbox-error .checkbox-custom {
  border-color: var(--checkbox-border-error);
}

.checkbox-label {
  color: var(--checkbox-label-color);
  font-size: 16px;
  line-height: 1.5;
  user-select: none;
}
</style>
</head>
<body>
  <label class="checkbox-container">
    <input type="checkbox" class="checkbox-native" />
    <span class="checkbox-custom">
      <svg class="checkbox-icon" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
        <polyline points="20 6 9 17 4 12"></polyline>
      </svg>
    </span>
    <span class="checkbox-label">Ich akzeptiere die AGB</span>
  </label>
</body>
</html>

```

## Design Tokens

Token	Wert	Verwendung
checkbox.size.default	20px	Standard Checkbox-Größe
checkbox.size.large	24px	Mobile-optimierte Größe
checkbox.borderRadius	4px	Eckenradius
checkbox.borderWidth	2px	Rahmenbreite
checkbox.background.default	{neutral.white}	Hintergrund unchecked
checkbox.background.hover	{neutral.white}	Hintergrund unchecked hover
checkbox.background.checked	{hydrophon.blau.500}	Hintergrund checked
checkbox.background.checkedHover	{hydrophon.blau.600}	Hintergrund checked hover
checkbox.background.disabled	{neutral.50}	Hintergrund disabled unchecked
checkbox.background.checkedDisabled	{neutral.300}	Hintergrund disabled checked
checkbox.border.default	{neutral.300}	Rahmen Standard
checkbox.border.hover	{neutral.400}	Rahmen Hover
checkbox.border.checked	{hydrophon.blau.500}	Rahmen Checked
checkbox.border.focus	{hydrophon.blau.500}	Rahmen Focus
checkbox.border.error	{color.error}	Rahmen Error
checkbox.border.disabled	{neutral.200}	Rahmen Disabled
checkbox.icon.color	{neutral.white}	Icon Farbe
checkbox.icon.size	14px	Icon Größe
checkbox.focus.outlineWidth	2px	Focus Outline Breite
checkbox.focus.outlineOffset	2px	Focus Outline Abstand
checkbox.focus.outlineColor	{hydrophon.blau.300}	Focus Outline Farbe
checkbox.label.gap	{spacing.2} (8px)	Abstand zu Label
checkbox.label.color	{neutral.700}	Label Farbe
checkbox.label.colorDisabled	{neutral.400}	Label Farbe Disabled

## Anti-Patterns

### Vermeiden

#### 1. `display: none` für natives Input

- Entfernt Element komplett aus Accessibility Tree
- Screenreader können Checkbox nicht erkennen
- Lösung: Verwende `opacity: 0 + position: absolute`

#### 2. Nur `<div>` mit `onClick` ohne Checkbox-Semantik

```
<!-- FALSCH -->
<div onclick="toggle()>Option auswählen</div>
```

- Keine native Checkbox-Rolle
- Tastatur-Navigation funktioniert nicht
- Screenreader erkennen keine Checkbox
- **Lösung:** Verwende natives `<input type="checkbox">`

### 3. Fehlender Focus-Indikator

```
/* FALSCH */
.checkbox:focus {
  outline: none; /* NIEMALS ohne Ersatz! */
}
```

- WCAG 2.4.7 Verletzung
- Tastatur-Nutzer verlieren Orientierung
- **Lösung:** Immer sichtbaren Focus-Indikator bereitstellen

### 4. Nur Farbe zur Unterscheidung checked/unchecked

- Farbblinde Nutzer können Zustand nicht erkennen
- WCAG 1.4.1 Verletzung
- **Lösung:** Kombiniere Farbe mit Icon (Checkmark bei checked)

### 5. Zu kleine Touch-Targets

- Checkbox 16px ohne Label-Klickfläche
- Mobile Nutzer verfehlten Target
- **Lösung:** Minimum 20px Checkbox + Label klickbar, oder 24px Large Size

### 6. Label nicht mit Checkbox verknüpft

```
<!-- FALSCH -->
<input type="checkbox" id="terms" />
<span>AGB akzeptieren</span> <!-- kein <label> -->
```

- Label-Klick aktiviert Checkbox nicht
- Reduzierte Klickfläche
- **Lösung:** Verwende `<label>` mit `for` oder umschließend

## Best Practices

1. **Verwende natives `<input type="checkbox">`** - Native Semantik ist allen Custom-Lösungen überlegen
2. **Opacity-Technik statt `display:none`** - Erhält Accessibility und Tastatur-Navigation
3. **Check-Icon bei checked-Zustand** - Nicht nur Farbwechsel für Barrierefreiheit
4. **Label immer klickbar** - Vergrößert Klickfläche und erfüllt WCAG Touch-Target
5. **:focus-visible verwenden** - Focus nur bei Tastatur-Navigation, nicht bei Maus
6. **Fieldset für Gruppen** - Semantisch korrekt und Screenreader-freundlich
7. **Error-Messages erklärend** - "Bitte akzeptieren Sie die AGB" statt nur "Fehler"

8. **aria-describedby** für Helper/Error - Verknüpft zusätzliche Infos mit Checkbox

---

Komponente: Checkbox (FORM-04) Phase: 03-forms-a-data-input Version: 1.0

---

# Radio Button

---

Ermöglicht Auswahl genau **einer** Option aus einer Gruppe. Radio Buttons sind für exklusive Entscheidungen - die Auswahl einer Option hebt automatisch alle anderen auf.

## Übersicht

### Anatomie

Ein Radio Button besteht aus folgenden Elementen:

- **Native Input** - Natives `<input type="radio">` Element (visuell versteckt, Accessibility erhalten)
- **Custom Visual** - Gestylter Kreis-Container, der das native Element überlagert
- **Inner Dot** - Weißer gefüllter Kreis (50% der Gesamtgröße) bei ausgewähltem Zustand
- **Label-Text** - Beschreibender Text neben dem Radio Button
- **Helper Text** (optional) - Zusätzliche Erklärung unter der Option
- **Error Message** (optional) - Validierungsfehlermeldung für die gesamte Gruppe

## Verwendung

Radio Buttons eignen sich für:

- **Exklusive Auswahl** - Nur eine Option aus mehreren möglich (z.B. Zahlungsmethode)
- **Versandoptionen** - Standard, Express, Overnight
- **Einstellungen** - Ja/Nein, Klein/Mittel/Groß
- **Prioritäten** - Niedrig, Normal, Hoch
- **Sortierung** - Aufsteigend, Absteigend

Nicht verwenden für:

- Mehrfachauswahl (0 bis N Optionen möglich) → verwende Checkboxen
- Ein/Aus-Schalter mit sofortiger Wirkung → verwende Toggle Switch
- Sehr viele Optionen (>7) → verwende Select/Dropdown

## Unterschied zu Checkbox

Kriterium	Radio Button	Checkbox
Auswahl	Exakt <b>eine</b> Option	Null, <b>eine</b> oder <b>mehrere</b> Optionen
Verhalten	Auswahl hebt vorherige auf	Jede unabhängig an/abwählbar
Verwendung	Mutuell exklusive Werte	Unabhängige Ja/Nein-Entscheidungen
Beispiel	Versandart: Standard ODER Express	Newsletter: News UND Events UND Angebote
Pflicht	Immer eine Option ausgewählt	Keine Auswahl möglich

**Wichtig:** Radio Buttons sollten **immer in Gruppen** erscheinen (minimum 2 Optionen). Ein einzelner Radio Button ergibt keinen Sinn, da er nicht abgewählt werden kann.

## Größen

### Default (20px)

- Radio: 20x20px Kreis
- Inner Dot: 10px (50% der Gesamtgröße)
- Verwendung: Standard für die meisten Formulare
- Desktop und Mobile geeignet

### Large (24px)

- Radio: 24x24px Kreis
- Inner Dot: 12px (50% der Gesamtgröße)
- Verwendung: Mobile-optimierte Interfaces, verbesserte Touch-Targets
- Erfüllt WCAG AAA 44px Empfehlung in Kombination mit Label-Klickfläche

## Zustände

### Unselected Default

- Background: neutral.white
- Border: neutral.300 (2px)
- Inner Dot: nicht sichtbar
- Visuell: Leerer weißer Kreis mit grauem Rahmen

### Unselected Hover

- Background: neutral.white
- Border: neutral.400 (dunkler)
- Inner Dot: nicht sichtbar
- Feedback: Subtile Verdunklung des Rahmens

### Selected

- Background: hydrophon.blau.500
- Border: hydrophon.blau.500
- Inner Dot: Weißer Kreis (50% der Gesamtgröße)
- Visuell: Blauer Kreis mit weißem Punkt in der Mitte

### Selected Hover

- Background: hydrophon.blau.600 (dunkler)
- Border: hydrophon.blau.600
- Inner Dot: Weißer Kreis
- Feedback: Verdunklung bei Hover über ausgewählten Radio Button

## Focus

- Outline: 2px hydrophon.blau.300
- Offset: 2px

- Nur bei `:focus-visible` (Tastatur-Navigation)
- Kombiniert mit aktuellem Zustand (selected/unselected)

**Wichtig:** Focus-Indikator erscheint NUR bei Tastatur-Navigation, nicht bei Maus-Klick. Dies verhindert visuelles Rauschen und erfüllt gleichzeitig WCAG 2.4.7 (Focus sichtbar).

## Error

- Border: `color.error` (rot)
- Error-Message unter der gesamten Radio-Gruppe (nicht pro Option)
- Alle Radios in der Gruppe erhalten roten Border
- Bei Pflichtfeld-Validierung: "Bitte wählen Sie eine Option aus"

## Disabled (Unselected)

- Background: `neutral.50` (leicht grau)
- Border: `neutral.200` (gedämpft)
- Label: `neutral.400` (gedämpft)
- Cursor: `not-allowed`
- Nicht interaktiv, nicht fokussierbar

## Disabled (Selected)

- Background: `neutral.300` (gedämpftes Grau)
- Border: `neutral.300`
- Inner Dot: Weißer Kreis (reduzierter Kontrast)
- Label: `neutral.400`
- Zeigt vorherige Auswahl, aber nicht änderbar

## Radio-Gruppe

Radio Buttons funktionieren nur in Gruppen korrekt. Alle Radios mit gleichem `name` Attribut gehören zu einer Gruppe.

### Tastaturnavigation (native)

Native `<input type="radio">` Elemente haben spezielle Tastatur-Navigation:

- **Tab** - Fokussiert die aktuell ausgewählte Radio in der Gruppe (oder erste, falls keine ausgewählt)
- **Pfeiltasten ( $\uparrow/\downarrow$ )** - Wechselt und wählt automatisch die nächste/vorherige Option
- **Pfeiltasten ( $\leftarrow/\rightarrow$ )** - Wie  $\uparrow/\downarrow$  (funktioniert für horizontal angeordnete Gruppen)
- **Space** - Wählt die aktuell fokussierte Option aus

**Wichtig:** Diese native Tastatur-Navigation funktioniert NUR mit nativen `<input type="radio">` Elementen. Custom div-basierte Radios ohne natives Input verlieren diese Funktionalität.

### Fieldset-Pattern

Verwende `<fieldset>` und `<legend>` für Radio-Gruppen:

```

<fieldset>
  <legend>Versandart wählen <span aria-label="Pflichtfeld">*</span></legend>

  <label class="radio-container">
    <input type="radio" name="shipping" value="standard" class="radio-native" />
    <span class="radio-custom">
      <span class="radio-dot"></span>
    </span>
    <span class="radio-label">Standard (3-5 Werkstage) - Kostenlos</span>
  </label>

  <label class="radio-container">
    <input type="radio" name="shipping" value="express" class="radio-native" />
    <span class="radio-custom">
      <span class="radio-dot"></span>
    </span>
    <span class="radio-label">Express (1-2 Werkstage) - 9,90 €</span>
  </label>

  <label class="radio-container">
    <input type="radio" name="shipping" value="overnight" class="radio-native" />
    <span class="radio-custom">
      <span class="radio-dot"></span>
    </span>
    <span class="radio-label">Overnight (nächster Werktag) - 19,90 €</span>
  </label>
</fieldset>

```

#### Wichtig:

- Gleicher `name`-Wert - Alle Radios in einer Gruppe müssen den gleichen `name` haben
- Unterschiedlicher `value`-Wert - Jedes Radio hat einen eindeutigen `value`
- `<legend>` beschreibt die Gruppe - Screenreader lesen: "Versandart wählen, Radio Group"
- `<fieldset>` gruppierter semantisch - Visuell und für Accessibility

## Orientierung

### Vertikal (Standard)

- Optionen untereinander angeordnet
- Verwendet für: 3+ Optionen, komplexe Labels
- Spacing: `{spacing.3}` (12px) zwischen Optionen

### Horizontal

- Optionen nebeneinander angeordnet
- Verwendet für: 2-4 kurze Optionen
- Spacing: `{spacing.4}` (16px) zwischen Optionen
- **Maximum: 3-4 Optionen** - Sonst horizontal scrolling nötig

```

/* Vertikale Gruppe (Standard) */
.radio-group {
  display: flex;
  flex-direction: column;
  gap: var(--radio-group-gap); /* 12px */
}

/* Horizontale Gruppe */
.radio-group-horizontal {
  display: flex;
  flex-direction: row;
  gap: var(--spacing-4); /* 16px */
  flex-wrap: wrap; /* Bei schmalen Viewports umbrechen */
}

```

## Custom-Styling-Pattern

### Accessibility-konformes Verstecken

Gleiche Technik wie bei Checkboxen - das native `<input type="radio">` Element MUSS im DOM verbleiben:

**RICHTIG:** `opacity: 0` behält Accessibility

```

.radio-native {
  opacity: 0;
  position: absolute;
  width: 1px;
  height: 1px;
  pointer-events: none;
}

```

Diese Technik:

- Entfernt visuell das native Element
- Behält es im Accessibility Tree (Screenreader erkennen es)
- Erhält Tastatur-Navigation (Pfeiltasten funktionieren)
- Erhält native Radio-Gruppen-Logik (nur eine Option ausgewählt)

**FALSCH:** `display: none` entfernt aus Accessibility Tree

```

.radio-native {
  display: none; /* NIEMALS! */
}

```

Diese Technik:

- Entfernt Element komplett aus DOM
- Screenreader können es nicht erkennen
- Pfeil-Navigation funktioniert nicht
- Radio-Gruppen-Logik bricht
- WCAG 4.1.2 Verletzung

## Focus-Weiterleitung und Checked-State

```

/* Focus auf nativem Input stylt Custom Visual */
.radio-native:focus-visible + .radio-custom {
  outline: 2px solid var(--radio-focus-outline-color);
  outline-offset: 2px;
}

/* Checked-State - Kreis gefüllt */
.radio-native:checked + .radio-custom {
  background-color: var(--radio-background-checked);
  border-color: var(--radio-border-checked);
}

/* Inner Dot erscheint nur bei checked */
.radio-dot {
  opacity: 0;
  width: var(--radio-dot-size);
  height: var(--radio-dot-size);
  border-radius: 50%;
  background-color: var(--radio-dot-color);
  transition: opacity 150ms ease-in-out;
}

.radio-native:checked + .radio-custom .radio-dot {
  opacity: 1;
}

/* Checked + Hover kombiniert */
.radio-native:checked + .radio-custom:hover {
  background-color: var(--radio-background-checked-hover);
}

/* Disabled State */
.radio-native:disabled + .radio-custom {
  background-color: var(--radio-background-disabled);
  border-color: var(--radio-border-disabled);
  cursor: not-allowed;
}

/* Disabled + Checked kombiniert */
.radio-native:disabled:checked + .radio-custom {
  background-color: var(--radio-background-checked-disabled);
}

.radio-native:disabled ~ .radio-label {
  color: var(--radio-label-color-disabled);
}

```

## Accessibility

### WCAG Anforderungen

#### 4.1.2 Name, Rolle, Wert

- Native `<input type="radio">` behält automatisch Radio-Rolle
- Screenreader kündigen korrekt an: "radio button, not checked" / "radio button, checked"
- Keine zusätzlichen ARIA-Attribute nötig (außer `aria-describedby` für Errors)

#### 2.4.3 Focus-Reihenfolge

- Tab wechselt zwischen Radio-Gruppen (nicht zwischen einzelnen Radios)
- Pfeiltasten navigieren innerhalb der Gruppe
- Focus springt zur aktuell ausgewählten Option (oder erste, falls keine ausgewählt)

#### 3.3.2 Labels oder Instructions

- `<legend>` beschreibt die Gruppe ("Versandart wählen")
- Jedes `<label>` beschreibt die einzelne Option ("Standard", "Express")
- Screenreader lesen: "Versandart wählen, Radio Group, Standard, radio button, not checked"

#### 1.4.3 Kontrast

- Text-Kontrast: 4.5:1 (Label auf weißem Hintergrund)
- UI-Komponenten-Kontrast: 3:1 (Border auf weißem Hintergrund)
- Selected-State: Blau mit weißem Dot für maximalen Kontrast

#### 2.5.5 Touch-Target-Größe (WCAG 2.2)

- Minimum: 44x44px (AAA-Empfehlung)
- Default Radio (20px) + Label-Klickfläche erfüllt dies
- Large Radio (24px) für reine Icon-Darstellung

### Native Keyboard-Patterns

Radio Buttons innerhalb einer Gruppe (gleicher `name`) haben spezielle Navigation:

#### Erste Tabulator-Taste:

- Fokussiert aktuell ausgewählte Radio in der Gruppe
- Falls keine ausgewählt: Fokussiert erste Radio

#### Pfeiltasten (`↑/↓` oder `←/→`):

- Wechselt zur nächsten/vorherigen Option
- Wählt automatisch die neue Option aus (checked-State ändert sich)
- Bei letzter Option: Springt zur ersten (wrap around)

#### Space-Taste:

- Wählt aktuell fokussierte Option aus
- Überschreibt vorherige Auswahl

Diese Navigation funktioniert NUR mit nativen `<input type="radio">` Elementen!

### HTML-Struktur

Empfohlene Struktur:

```

<fieldset>
  <legend>Zahlungsmethode auswählen</legend>

  <label class="radio-container">
    <input type="radio" name="payment" value="invoice" class="radio-native" checked />
    <span class="radio-custom">
      <span class="radio-dot"></span>
    </span>
    <span class="radio-label">Rechnung</span>
  </label>

  <label class="radio-container">
    <input type="radio" name="payment" value="card" class="radio-native" />
    <span class="radio-custom">
      <span class="radio-dot"></span>
    </span>
    <span class="radio-label">Kreditkarte</span>
  </label>

  <label class="radio-container">
    <input type="radio" name="payment" value="paypal" class="radio-native" />
    <span class="radio-custom">
      <span class="radio-dot"></span>
    </span>
    <span class="radio-label">PayPal</span>
  </label>
</fieldset>

```

Mit Helper Text pro Option:

```

<label class="radio-container">
  <input type="radio" name="plan" value="basic" class="radio-native" aria-describedby="basic-helper" />
  <span class="radio-custom">
    <span class="radio-dot"></span>
  </span>
  <div>
    <span class="radio-label">Basic Plan</span>
    <p id="basic-helper" class="radio-helper">Für kleine Teams (bis 5 Nutzer)</p>
  </div>
</label>

```

## Error-Handling

### Pflichtfeld-Validierung

Radio-Gruppen mit `required` Attribut:

```

<fieldset aria-describedby="shipping-error">
  <legend>Versandart wählen *</legend>

  <label class="radio-container radio-error">
    <input type="radio" name="shipping" value="standard" class="radio-native" required />
    <span class="radio-custom">
      <span class="radio-dot"></span>
    </span>
    <span class="radio-label">Standard</span>
  </label>

  <label class="radio-container radio-error">
    <input type="radio" name="shipping" value="express" class="radio-native" required />
    <span class="radio-custom">
      <span class="radio-dot"></span>
    </span>
    <span class="radio-label">Express</span>
  </label>

  <p id="shipping-error" class="radio-error-message">
    Bitte wählen Sie eine Versandart aus
  </p>
</fieldset>

```

#### Wichtig:

- `required` auf allen Radios in der Gruppe (nicht nur erstem)
- Error-Message unter der gesamten Gruppe
- `aria-describedby` auf `<fieldset>` verknüpft Error mit Gruppe
- Border-Color aller Radios wird rot

#### Styling bei Error

```

/* Error-State für alle Radios in Gruppe */
.radio-container.radio-error .radio-custom {
  border-color: var(--radio-border-error);
}

/* Error-Message */
.radio-error-message {
  color: var(--input-error-color);
  font-size: var(--input-error-font-size);
  margin-top: var(--input-error-margin-top);
  display: flex;
  align-items: center;
  gap: 4px;
}

/* Optional: Error-Icon vor Message */
.radio-error-message::before {
  content: "⚠";
  font-size: 16px;
}

```

## Code-Beispiel

Vollständiges HTML + CSS Beispiel:

```
<!DOCTYPE html>
<html lang="de">
<head>
  <style>
    fieldset {
      border: none;
      padding: 0;
      margin: 0;
      display: flex;
      flex-direction: column;
      gap: var(--radio-group-gap);
    }

    legend {
      font-weight: 500;
      font-size: 14px;
      color: var(--neutral-700);
      margin-bottom: 8px;
    }

    .radio-container {
      display: inline-flex;
      align-items: flex-start;
      gap: var(--radio-label-gap);
      cursor: pointer;
      position: relative;
    }

    .radio-native {
      opacity: 0;
      position: absolute;
      width: 1px;
      height: 1px;
      pointer-events: none;
    }

    .radio-custom {
      display: flex;
      align-items: center;
      justify-content: center;
      width: var(--radio-size-default);
      height: var(--radio-size-default);
      border: var(--radio-border-width) solid var(--radio-border-default);
      border-radius: var(--radio-border-radius);
      background-color: var(--radio-background-default);
      transition: all 150ms ease-in-out;
      flex-shrink: 0;
    }

    .radio-dot {
      width: var(--radio-dot-size);
      height: var(--radio-dot-size);
      border-radius: 50%;
      background-color: var(--radio-dot-color);
      opacity: 0;
      transition: opacity 150ms ease-in-out;
    }
  </style>
</head>
<body>
  <form>
    <legend>Legend</legend>
    <div class="radio-container">
      <input type="radio" class="radio-native" checked="" />
      <div class="radio-custom">
        <div class="radio-dot">
```

```

/* Hover */
.radio-custom:hover {
  border-color: var(--radio-border-hover);
}

/* Checked */
.radio-native:checked + .radio-custom {
  background-color: var(--radio-background-checked);
  border-color: var(--radio-border-checked);
}

.radio-native:checked + .radio-custom .radio-dot {
  opacity: 1;
}

/* Checked + Hover */
.radio-native:checked + .radio-custom:hover {
  background-color: var(--radio-background-checked-hover);
}

/* Focus */
.radio-native:focus-visible + .radio-custom {
  outline: var(--radio-focus-outline-width) solid var(--radio-focus-outline-color);
  outline-offset: var(--radio-focus-outline-offset);
}

/* Disabled */
.radio-native:disabled + .radio-custom {
  background-color: var(--radio-background-disabled);
  border-color: var(--radio-border-disabled);
  cursor: not-allowed;
}

.radio-native:disabled:checked + .radio-custom {
  background-color: var(--radio-background-checked-disabled);
}

.radio-native:disabled ~ .radio-label {
  color: var(--radio-label-color-disabled);
}

/* Error */
.radio-container.radio-error .radio-custom {
  border-color: var(--radio-border-error);
}

.radio-label {
  color: var(--radio-label-color);
  font-size: 16px;
  line-height: 1.5;
  user-select: none;
}
</style>
</head>
<body>
<fieldset>
<legend>Versandart auswählen</legend>

```

```
<label class="radio-container">
  <input type="radio" name="shipping" value="standard" class="radio-native" checked />
  <span class="radio-custom">
    <span class="radio-dot"></span>
  </span>
  <span class="radio-label">Standard (3-5 Werkstage) - Kostenlos</span>
</label>

<label class="radio-container">
  <input type="radio" name="shipping" value="express" class="radio-native" />
  <span class="radio-custom">
    <span class="radio-dot"></span>
  </span>
  <span class="radio-label">Express (1-2 Werkstage) - 9,90 €</span>
</label>

<label class="radio-container">
  <input type="radio" name="shipping" value="overnight" class="radio-native" />
  <span class="radio-custom">
    <span class="radio-dot"></span>
  </span>
  <span class="radio-label">Overnight (nächster Werktag) - 19,90 €</span>
</label>
</fieldset>
</body>
</html>
```

## Design Tokens

Token	Wert	Verwendung
radio.size.default	20px	Standard Radio-Größe
radio.size.large	24px	Mobile-optimierte Größe
radio.borderRadius	50%	Perfekter Kreis
radio.borderWidth	2px	Rahmenbreite
radio.background.default	{neutral.white}	Hintergrund unselected
radio.background.hover	{neutral.white}	Hintergrund unselected hover
radio.background.checked	{hydrophon.blau.500}	Hintergrund selected
radio.background.checkedHover	{hydrophon.blau.600}	Hintergrund selected hover
radio.background.disabled	{neutral.50}	Hintergrund disabled unselected
radio.background.checkedDisabled	{neutral.300}	Hintergrund disabled selected
radio.border.default	{neutral.300}	Rahmen Standard
radio.border.hover	{neutral.400}	Rahmen Hover
radio.border.checked	{hydrophon.blau.500}	Rahmen Selected
radio.border.focus	{hydrophon.blau.500}	Rahmen Focus
radio.border.error	{color.error}	Rahmen Error
radio.border.disabled	{neutral.200}	Rahmen Disabled
radio.dot.size	10px	Inner Dot Größe Default
radio.dot.sizeLarge	12px	Inner Dot Größe Large
radio.dot.color	{neutral.white}	Inner Dot Farbe
radio.focus.outlineWidth	2px	Focus Outline Breite
radio.focus.outlineOffset	2px	Focus Outline Abstand
radio.focus.outlineColor	{hydrophon.blau.300}	Focus Outline Farbe
radio.label.gap	{spacing.2} (8px)	Abstand zu Label
radio.label.color	{neutral.700}	Label Farbe
radio.label.colorDisabled	{neutral.400}	Label Farbe Disabled
radio.group.gap	{spacing.3} (12px)	Abstand zwischen Optionen
radio.group.orientation	vertical	Standard-Orientierung

## Anti-Patterns

### Vermeiden

1. Custom div-Radios ohne native input

```
<!-- FALSCH -->
<div class="radio" onclick="select('option1')">Option 1</div>
```

- Keine native Radio-Rolle
- Pfeil-Navigation funktioniert nicht
- Screenreader erkennen keine Radio-Gruppe
- Radio-Gruppen-Logik (nur eine Option) fehlt
- **Lösung:** Verwende natives `<input type="radio">`

## 2. Einzelne Radio ohne Gruppe

```
<!-- FALSCH -->
<input type="radio" name="single" /> Einzelne Option
```

- Radio kann nicht abgewählt werden (stuck in checked-State)
- Ergibt semantisch keinen Sinn
- **Lösung:** Verwende Checkbox für Ja/Nein oder Toggle für Ein/Aus

## 3. Fehlende Legende für Gruppe

```
<!-- FALSCH -->
<div>
  <input type="radio" name="payment" /> Rechnung
  <input type="radio" name="payment" /> Kreditkarte
</div>
```

- Screenreader wissen nicht, was die Gruppe beschreibt
- WCAG 3.3.2 Verletzung
- **Lösung:** Verwende `<fieldset>` + `<legend>`

## 4. Unterschiedliche `name` Werte in einer Gruppe

```
<!-- FALSCH -->
<input type="radio" name="option1" /> Option 1
<input type="radio" name="option2" /> Option 2
```

- Browser behandelt sie als separate Gruppen
- Mehrere Optionen können gleichzeitig ausgewählt sein
- **Lösung:** Alle Radios in einer Gruppe brauchen gleichen `name`

## 5. `display: none` für natives Input

- Entfernt Element aus Accessibility Tree
- Pfeil-Navigation bricht
- **Lösung:** Verwende `opacity: 0` + `position: absolute`

## 6. Label nicht mit Radio verknüpft

```
<!-- FALSCH -->
<input type="radio" id="option1" name="choice" />
<span>Option 1</span> <!-- kein <label> -->
```

- Label-Klick aktiviert Radio nicht
- Reduzierte Klickfläche
- Lösung: Verwende `<label>` mit `for` oder umschließend

#### 7. Zu viele Optionen ohne Gruppierung

- 10+ Radio Buttons untereinander
- Überwältigend für Nutzer
- Lösung: Verwende Select/Dropdown oder gruppieren in Sub-Kategorien

## Best Practices

1. Verwende natives `<input type="radio">` - Native Semantik und Tastatur-Navigation unverzichtbar
2. Opacity-Technik statt `display:none` - Erhält Accessibility und Pfeil-Navigation
3. Immer in Gruppen (minimum 2 Optionen) - Einzelne Radios ergeben keinen Sinn
4. Gleicher `name` für Gruppe - Browser erkennt dann exklusive Auswahl
5. Fieldset + Legend verwenden - Semantisch korrekt und Screenreader-freundlich
6. Inner Dot statt Icon - Einfacher zu stylen, konsistent mit nativen Radios
7. Label immer klickbar - Vergrößert Klickfläche (WCAG Touch-Target)
8. Eine Option vorausgewählt - Verhindert ungewollte Submissions ohne Auswahl
9. `:focus-visible` verwenden - Focus nur bei Tastatur, nicht bei Maus
10. Error-Messages erklärend - "Bitte wählen Sie eine Versandart" statt nur "Fehler"
11. Maximum 7 Optionen - Mehr Optionen → Select/Dropdown verwenden

## Wann Radio, wann Checkbox, wann Select?

Anzahl Optionen	Typ	Auswahl	Komponente
2-7	Exklusiv (eine)	Sichtbar	Radio Button
2-7	Mehrfach (0-N)	Sichtbar	Checkbox
8+	Exklusiv (eine)	Ausklappbar	Select/Dropdown
8+	Mehrfach (0-N)	Ausklappbar	Multi-Select
1	Ja/Nein	Sofort wirksam	Toggle Switch
1	Ja/Nein	Bei Submit wirksam	Checkbox

Komponente: Radio Button (FORM-05) Phase: 03-forms-a-data-input Version: 1.0

# Labels und Helper Text

---

Beschriftungen und Hilfestellungen fuer Formularfelder.

## Labels

### Grundprinzipien

- Jedes Eingabefeld MUSS ein sichtbares Label haben
- Labels stehen UEBER dem Eingabefeld (nicht links, nicht als Placeholder)
- Labels beschreiben den erwarteten Inhalt praezise
- Kurz und praegnant (max 3-4 Woerter idealerweise)

### Technische Verknuepfung

Es gibt zwei HTML-Pattern fuer Label-Input-Verknuepfungen:

#### Pattern 1: for/id Verknuepfung (Empfohlen)

```
<label for="email">E-Mail-Adresse</label>
<input type="email" id="email" />
```

#### Pattern 2: Umschliessendes Label

```
<label>
  E-Mail-Adresse
  <input type="email" />
</label>
```

**Empfehlung:** for/id ist expliziter und ermoeglicht flexibleres Layout. Umschliessende Labels koennen bei komplexen Layouts zu Styling-Problemen fuehren.

### Warum Labels ueber dem Input?

#### Vorteile:

- Konsistente vertikale Leserichtung (natuerlicher Lesefluss)
- Mobile-optimiert (bei schmalen Viewports kein Platz fuer Label links)
- Bessere Skalierung (Labels koennen variabel lang sein)
- WCAG-konform (Screenreader lesen von oben nach unten)
- Einfacheres Responsive Design

#### Vermeiden:

- Labels links vom Input (bricht auf Mobile)
- Labels rechts vom Input (verwirrt die Lesereihenfolge)

- Nur Placeholder ohne Label (nicht barrierefrei)

## Styling

Labels verwenden folgende Token:

- **Schriftgroesse:** {input.label.fontSize} → 14px
- **Schriftgewicht:** {input.label.fontWeight} → 500 (medium)
- **Farbe:** {input.label.color.default} → neutral.700
- **Abstand zum Input:** {input.label.marginBottom} → 8px

```
label {
  display: block;
  font-size: var(--input-label-font-size);
  font-weight: var(--input-label-font-weight);
  color: var(--input-label-color-default);
  margin-bottom: var(--input-label-margin-bottom);
}
```

## Label-Text-Empfehlungen

Gut:

- "E-Mail-Adresse"
- "Telefonnummer"
- "Lieferdatum"
- "Nachricht"

Vermeiden:

- "Bitte geben Sie Ihre E-Mail-Adresse ein" (zu lang, gehört in Helper Text)
- "Email:" (Doppelpunkt unnötig, nicht-deutscher Begriff)
- "Ihre Nachricht an uns" (zu umgangssprachlich für B2B)

## Pflichtfeld-Kennzeichnung

### Visueller Indikator

Pflichtfelder werden mit einem roten Asterisk (\*) nach dem Label-Text gekennzeichnet:

```
<label for="name">
  Name
  <span class="required" aria-label="Pflichtfeld">*</span>
</label>
<input type="text" id="name" aria-required="true" required />
```

### Styling des Asterisk

- **Farbe:** {input.required.color} → color.error (rot)
- **Symbol:** {input.required.symbol} → "\*"
- **Position:** Direkt nach dem Label-Text mit kleinem Abstand (2px margin-left)

```
.required {
  color: var(--input-required-color);
  margin-left: 2px;
}
```

## Screenreader-Unterstuetzung

Der Asterisk benoetigt `aria-label="Pflichtfeld"`, damit Screenreader-Nutzer die Information erhalten:

```
<span class="required" aria-label="Pflichtfeld">*</span>
```

Alternativ kann die Information im Label selbst stehen:

```
<label for="name">
  Name <span aria-label="Pflichtfeld">*</span>
</label>
```

## Legende am Formularanfang

Wenn mehrere Pflichtfelder im Formular sind, fuge eine Legende am Anfang hinzu:

```
<form>
  <p class="form-legend">Felder mit <span class="required">*</span> sind Pflichtfelder</p>
  <!-- Formularfelder -->
</form>
```

## HTML-Attribute fuer Pflichtfelder

Pflichtfelder benoetigen ZWEI Attribute:

1. `required` - Native HTML5 Validierung
2. `aria-required="true"` - Screenreader-Ankuendigung

```
<input
  type="text"
  id="name"
  required
  aria-required="true"
/>
```

Warum beide?

- `required` : Browser validiert vor Submit, zeigt native Fehlermeldung
- `aria-required="true"` : Screenreader kuendigt Pflichtfeld an BEVOR User eingibt

## Alternativer Ansatz: "(optional)" statt Asterisk

Wenn 80% oder mehr der Felder Pflichtfelder sind, ist es effizienter, die OPTIONALEN Felder zu markieren:

```
<!-- Statt Asterisk bei 8 von 10 Feldern... -->
<label for="name">Name <span class="required">*</span></label>
<label for="email">E-Mail <span class="required">*</span></label>
<label for="phone">Telefon <span class="required">*</span></label>
<!-- ... nur 2 mit "(optional)" kennzeichnen -->

<!-- Besser: -->
<label for="name">Name</label>
<label for="email">E-Mail</label>
<label for="phone">Telefon</label>
<label for="notes">Notizen <span class="optional">(optional)</span></label>
<label for="fax">Fax <span class="optional">(optional)</span></label>
```

**Vorteil:** Weniger visuelle Unruhe, Fokus auf das Wesentliche.

## Error-State für Labels

Wenn ein Feld einen Fehler hat, kann das Label ebenfalls rot eingefärbt werden:

```
<label for="email" class="error">
  E-Mail-Adresse
  <span class="required" aria-label="Pflichtfeld">*</span>
</label>
<input
  type="email"
  id="email"
  aria-invalid="true"
  aria-describedby="email-error"
/>
<span id="email-error" role="alert">
  Bitte gib eine gültige E-Mail-Adresse ein
</span>
```

```
label.error {
  color: var(--input-label-color-error);
}
```

## Helper Text

### Zweck

Helper Text erklärt das Eingabefeld BEVOR der User eingibt:

- **Eingabeformat:** "TT.MM.JJJJ" oder "z.B. 01.12.2024"
- **Beispielwerte:** "z.B. name@firma.de"
- **Zusätzliche Kontext-Information:** "Wir verwenden deine E-Mail nur zur Kontaktaufnahme"

- Einschraenkungen: "Mindestens 8 Zeichen, mindestens ein Grossbuchstabe"

## Position

Helper Text steht UNTER dem Eingabefeld:

```
[Label]
[Input Field]
[Helper Text]
```

## Abstand

- Abstand nach oben vom Input: `{input.helper.marginTop}` → 4px

```
.helper-text {
  margin-top: var(--input-helper-margin-top);
}
```

## Styling

- Schriftgroesse: `{input.helper.fontSize}` → 12px
- Farbe: `{input.helper.color}` → neutral.600
- Zeilenhoehe: 1.4 (fuer mehrzeiligen Text)

```
.helper-text {
  display: block;
  font-size: var(--input-helper-font-size);
  color: var(--input-helper-color);
  margin-top: var(--input-helper-margin-top);
  line-height: 1.4;
}
```

## Accessibility: aria-describedby

Helper Text MUSS mit dem Input via `aria-describedby` verknuepft werden:

```
<label for="password">Passwort</label>
<input
  type="password"
  id="password"
  aria-describedby="password-helper"
/>
<span id="password-helper" class="helper-text">
  Mindestens 8 Zeichen, mindestens ein Grossbuchstabe
</span>
```

## Warum wichtig?

- Screenreader lesen Helper Text VOR der Eingabe

- User wissen sofort, was erwartet wird
- Reduziert Fehlerquote

## Placeholder vs Helper Text

Eigenschaft	Placeholder	Helper Text
Sichtbarkeit	Verschwindet bei Eingabe	Bleibt sichtbar
Kontrast	Geringer Kontrast (neutral.400)	Voller Kontrast (neutral.600)
Screenreader	Nicht als Label erkannt	Mit aria-describedby verknuepft
WCAG-Konformitaet	Problematisch (zu geringer Kontrast)	Konform
Verwendung	Optionale Beispielwerte wenn Platz fehlt	Wichtige Informationen

**Empfehlung:** Helper Text fuer wichtige Informationen, Placeholder nur fuer Beispielwerte.

## Wann Placeholder verwenden?

Placeholder sind hilfreich fuer:

- **EingabefORMAT-Beispiele:** placeholder="+49 123 456789" fuer Telefonnummer
- **Suchfelder:** placeholder="Produkt suchen..." (wenn offensichtlich)
- **OPTIONALE Zusatzinfos:** placeholder="Optional: Firma" (aber besser im Label)

## Best Practice: Kombination

```
<label for="phone">Telefonnummer</label>
<input
  type="tel"
  id="phone"
  placeholder="+49 123 456789"
  aria-describedby="phone-helper"
/>
<span id="phone-helper" class="helper-text">
  Bitte mit Laendervorwahl
</span>
```

- **Placeholder:** Zeigt Beispielformat
- **Helper Text:** Erklaert Anforderung

## Helper Text bei Error State

Wenn ein Fehler auftritt, wird der Helper Text durch die Error Message ERSETZT (nicht zusaetzlich):

```

<!-- Normal: -->
<input aria-describedby="email-helper" />
<span id="email-helper" class="helper-text">
  z.B. name@firma.de
</span>

<!-- Error: -->
<input aria-describedby="email-error" aria-invalid="true" />
<span id="email-error" class="error-message" role="alert">
  Bitte gib eine gueltige E-Mail-Adresse ein
</span>

```

#### Warum ersetzen?

- Vermeidet visuelle Ueberladung
- Error Message ist wichtiger als Helper Text
- User fokussiert sich auf Fehlerkorrektur

**Ausnahme:** Wenn Helper Text kritische Informationen enthaelt (z.B. Passwort-Anforderungen), kann er zusaetzlich zur Error Message bleiben:

```

<input aria-describedby="password-helper password-error" aria-invalid="true" />
<span id="password-helper" class="helper-text">
  Mindestens 8 Zeichen, mindestens ein Grossbuchstabe
</span>
<span id="password-error" class="error-message" role="alert">
  Das Passwort ist zu kurz
</span>

```

## Accessibility

### WCAG 3.3.2: Labels oder Anweisungen

**Anforderung:** Labels oder Anweisungen werden bereitgestellt, wenn Inhalte Benutzereingaben erfordern.

#### Umsetzung:

- Jedes Eingabefeld hat ein sichtbares Label
- Labels sind programmatisch verknuepft (for/id oder umschliessend)
- Anweisungen (Helper Text) werden VOR der Eingabe bereitgestellt
- Pflichtfelder sind eindeutig gekennzeichnet

#### aria-describedby

aria-describedby verknuepft Input mit Helper Text und/oder Error Message:

#### Ein Element:

```

<input aria-describedby="helper" />
<span id="helper">Helper Text</span>

```

### Mehrere Elemente (space-separated):

```
<input aria-describedby="helper error" />
<span id="helper">Helper Text</span>
<span id="error">Error Message</span>
```

### Screenreader-Verhalten:

1. Label wird angekündigt
2. Input-Typ wird angekündigt
3. aria-describedby Elemente werden angekündigt
4. Input-Wert wird angekündigt (falls vorhanden)

### Fokus-Reihenfolge

Die Fokus-Reihenfolge für ein Standard-Formularfeld:

1. **Label:** Nicht fokussierbar, aber von Screenreader angekündigt
2. **Input:** Fokussierbar mit Tab
3. **Helper/Error Text:** Nicht fokussierbar, aber mit aria-describedby angekündigt

```
<label for="email">E-Mail-Adresse</label>           <!-- 1. Angekündigt -->
<input type="email" id="email" aria-describedby="help" /> <!-- 2. Fokussiert -->
<span id="help">z.B. name@firma.de</span>          <!-- 3. Angekündigt -->
```

### aria-required vs required

Beide Attribute sind wichtig:

Attribut	Zweck	Browser-Verhalten
required	Native HTML5 Validierung	Browser validiert vor Submit, zeigt Fehlermeldung
aria-required="true"	Screenreader-Ankündigung	Screenreader sagt "Pflichtfeld" beim Fokus

**Best Practice:** Beide verwenden.

```
<input type="text" required aria-required="true" />
```

## Layout-Pattern

### Standard-Feld-Struktur

```
<div class="form-field">
  <label for="field-id">
    Label-Text
    <span class="required" aria-label="Pflichtfeld">*</span>
  </label>
  <input
    type="text"
    id="field-id"
    aria-required="true"
    aria-describedby="field-helper"
    required
  />
  <span id="field-helper" class="helper-text">
    Helper Text
  </span>
</div>
```

### Styling mit Token-Referenzen

```
.form-field {
  margin-bottom: var(--form-spacing-field-to-field); /* 16px */
}

.form-field label {
  display: block;
  font-size: var(--input-label-font-size); /* 14px */
  font-weight: var(--input-label-font-weight); /* 500 */
  color: var(--input-label-color-default); /* neutral.700 */
  margin-bottom: var(--input-label-margin-bottom); /* 8px */
}

.form-field .required {
  color: var(--input-required-color); /* error */
  margin-left: 2px;
}

.form-field input {
  /* Input-Styles siehe text-input.md */
}

.form-field .helper-text {
  display: block;
  font-size: var(--input-helper-font-size); /* 12px */
  color: var(--input-helper-color); /* neutral.600 */
  margin-top: var(--input-helper-margin-top); /* 4px */
}
```

## Formulargruppen

Mehrere zusammenhaengende Felder koennen in einer Gruppe zusammengefasst werden:

```
<fieldset class="form-group">
  <legend>Adressdaten</legend>

  <div class="form-field">
    <label for="street">Strasse</label>
    <input type="text" id="street" />
  </div>

  <div class="form-field">
    <label for="city">Stadt</label>
    <input type="text" id="city" />
  </div>
</fieldset>
```

Abstand zwischen Gruppen:

```
.form-group {
  margin-bottom: var(--form-spacing-group-to-group); /* 24px */
}
```

## Anti-Patterns

### Vermeiden

#### 1. Placeholder als einziges Label

```
<!-- FALSCH -->
<input type="email" placeholder="E-Mail-Adresse" />

<!-- RICHTIG -->
<label for="email">E-Mail-Adresse</label>
<input type="email" id="email" placeholder="z.B. name@firma.de" />
```

**Warum?** Placeholder verschwindet bei Eingabe, geringer Kontrast, nicht WCAG-konform.

#### 2. Label rechts vom Input

```
<!-- FALSCH -->
<div style="display: flex;">
  <input type="checkbox" id="terms" />
  <label for="terms">Ich akzeptiere die AGB</label>
</div>
```

**Warum?** Das ist korrekt fuer Checkboxen, aber FALSCH fuer Text-Inputs. Bei Text-Inputs muss das Label UEBER dem Input stehen.

### 3. Floating Labels

```
<!-- VERMEIDEN -->
<div class="floating-label">
  <input type="text" id="email" placeholder=" " />
  <label for="email">E-Mail-Adresse</label>
</div>
```

**Warum?** Accessibility-Probleme, komplexe Implementierung, verwirrt Screenreader, schwierig fuer kognitive Einschraenkungen.

### 4. Label zu weit vom Input entfernt

```
<!-- FALSCH -->
<label for="email">E-Mail-Adresse</label>
<div style="margin-top: 40px;">
  <input type="email" id="email" />
</div>
```

**Warum?** User kann Label nicht mit Input assoziieren, besonders problematisch fuer kognitive Einschraenkungen.

### 5. Hilfetext nur in Tooltip versteckt

```
<!-- FALSCH -->
<label for="password">
  Passwort
  <button type="button" aria-label="Hilfe" title="Min. 8 Zeichen">?</button>
</label>
<input type="password" id="password" />

<!-- RICHTIG -->
<label for="password">Passwort</label>
<input type="password" id="password" aria-describedby="password-helper" />
<span id="password-helper" class="helper-text">
  Mindestens 8 Zeichen, mindestens ein Grossbuchstabe
</span>
```

**Warum?** Tooltip ist nicht keyboard-accessible, nicht von Screenreadern angekündigt, versteckt kritische Information.

## Code-Beispiele

### Einfaches Textfeld

```
<div class="form-field">
  <label for="name">Name</label>
  <input type="text" id="name" />
</div>
```

## Pflichtfeld mit Helper Text

```
<div class="form-field">
  <label for="email">
    E-Mail-Adresse
    <span class="required" aria-label="Pflichtfeld">*</span>
  </label>
  <input
    type="email"
    id="email"
    required
    aria-required="true"
    aria-describedby="email-helper"
  />
  <span id="email-helper" class="helper-text">
    Wir verwenden deine E-Mail nur zur Kontaktaufnahme
  </span>
</div>
```

## Feld mit Placeholder und Helper Text

```
<div class="form-field">
  <label for="phone">Telefonnummer</label>
  <input
    type="tel"
    id="phone"
    placeholder="+49 123 456789"
    aria-describedby="phone-helper"
  />
  <span id="phone-helper" class="helper-text">
    Bitte mit Laendervorwahl
  </span>
</div>
```

## Feld mit Error State

```
<div class="form-field">
  <label for="password" class="error">
    Passwort
    <span class="required" aria-label="Pflichtfeld">*</span>
  </label>
  <input
    type="password"
    id="password"
    required
    aria-required="true"
    aria-invalid="true"
    aria-describedby="password-helper password-error"
  />
  <span id="password-helper" class="helper-text">
    Mindestens 8 Zeichen, mindestens ein Grossbuchstabe
  </span>
  <span id="password-error" class="error-message" role="alert">
    Das Passwort ist zu kurz
  </span>
</div>
```

## Design Tokens

Alle Design Tokens fuer Labels und Helper Text:

Token	Wert	Verwendung
input.label.fontSize	14px	Label Schriftgroesse
input.label.fontWeight	500	Label Schriftgewicht (medium)
input.label.color.default	neutral.700	Label Farbe Standard
input.label.color.error	color.error	Label Farbe Error State
input.label.marginBottom	8px	Abstand Label → Input
input.helper.fontSize	12px	Helper Text Schriftgroesse
input.helper.color	neutral.600	Helper Text Farbe
input.helper.marginTop	4px	Abstand Input → Helper Text
input.required.color	color.error	Pflichtfeld-Asterisk Farbe (rot)
input.required.symbol	"*"	Pflichtfeld-Symbol
input.error.fontSize	12px	Error Message Schriftgroesse
input.error.color	color.error	Error Message Farbe
input.error.marginTop	4px	Abstand Input → Error Message
form.spacing.labelXInput	8px	Abstand Label → Input (alias für label.marginBottom)
form.spacing.inputToHelper	4px	Abstand Input → Helper/Error (alias für helper.marginTop)
form.spacing.fieldToField	16px	Abstand zwischen Formularfeldern
form.spacing.groupToGroup	24px	Abstand zwischen Formulargruppen

---

Siehe auch:

- [Text Input Dokumentation](#)
  - [Validation Dokumentation](#)
  - [Checkbox Dokumentation](#)
  - [Radio Button Dokumentation](#)
-

# Validierung und Fehlermeldungen

---

Patterns fuer Formular-Validierung mit klarem, hilfreichem Feedback.

## Validierungs-Strategie

### Progressive Validation (Empfohlen)

"Belohne frueh, bestrafe spaet"

Die beste UX-Strategie fuer Formular-Validierung basiert auf progressivem Feedback:

1. **Initiale Eingabe:** Keine Validierung waehrend User tippt (verhindert fruehe Fehler)
2. **Blur (Feld verlassen):** Erste Validierung wenn Feld verlassen wird
3. **Nach Fehler:** Echtzeit-Validierung bei jeder Aenderung (sofortiges Feedback bei Korrektur)
4. **Bei Korrektur:** Fehler verschwindet sofort wenn Input gueltig wird

## React Hook Form Konfiguration

```
import { useForm } from 'react-hook-form';

const form = useForm({
  mode: 'onBlur',           // Initiale Validierung bei blur
  reValidateMode: 'onChange' // Nach Fehler: live re-validieren
});
```

Warum dieses Pattern?

- **Verhindert Frustration:** User sieht keine Fehler waehrend er noch tippt
- **Sofortiges Feedback:** Nach einem Fehler bekommt User sofortige Bestaetigung bei Korrektur
- **Reduziert kognitive Last:** User kann sich auf Eingabe konzentrieren, nicht auf Fehlervermeidung
- **Bewahrt:** Nielsen Norman Group UX-Forschung empfiehlt diesen Ansatz

## Alternative: onSubmit (Nicht empfohlen fuer komplexe Formulare)

```
const form = useForm({
  mode: 'onSubmit' // Nur bei Submit validieren
});
```

Wann verwenden?

- Sehr einfache Formulare (1-2 Felder)
- Quick-Actions (z.B. Newsletter-Anmeldung)

Nachteile:

- User sieht Fehler erst am Ende

- Bei vielen Feldern: User muss alle Fehler auf einmal korrigieren
- Höhere Abbruchrate

### Alternative: onChange (Nicht empfohlen)

```
const form = useForm({
  mode: 'onChange' // Bei jedem Tastendruck validieren
});
```

#### Warum nicht?

- Zu früher Feedback ("E-Mail ungültig" nach dem ersten Buchstaben)
- Frustrierend für User
- Lenkt von Eingabe ab

## Error States

### Visueller Indikator

Fehlerhafte Felder werden durch Drei visuelle Indikatoren gekennzeichnet (WCAG 1.4.1: Fehler dürfen nicht nur durch Farbe kommuniziert werden):

1. **Rote Border:** {input.field.border.error} (color.error)
2. **Error-Icon:** AlertCircle (Lucide) im Input rechts (16px, error color)
3. **Error-Text:** Unter dem Input mit Erklärung

```
<div class="form-field form-field--error">
  <label for="email" class="error">
    E-Mail-Adresse
    <span class="required" aria-label="Pflichtfeld">*</span>
  </label>
  <div class="input-wrapper">
    <input
      type="email"
      id="email"
      aria-invalid="true"
      aria-describedby="email-error"
      class="input--error"
    />
    <svg class="input-icon input-icon--error" aria-hidden="true">
      <!-- AlertCircle Icon -->
    </svg>
  </div>
  <span id="email-error" class="error-message" role="alert">
    Bitte gib eine gültige E-Mail-Adresse ein (z.B. name@firma.de)
  </span>
</div>
```

## Token-Werte

```
.input--error {
  border-color: var(--input-field-border-error);      /* color.error */
}

.error-message {
  color: var(--input-error-color);                  /* color.error */
  font-size: var(--input-error-font-size);          /* 12px */
  margin-top: var(--input-error-margin-top);         /* 4px */
}

.input-icon--error {
  color: var(--input-error-color);                  /* color.error */
  width: 16px;
  height: 16px;
}
```

## WCAG-konforme Fehleranzeige

Fehler dürfen NICHT nur durch Farbe kommuniziert werden (WCAG 1.4.1):

Indikator	Zweck	Zielgruppe
Rote Border	Visueller Hinweis	Sehende User
Error-Icon	Form-basierter Hinweis	Farbenblinde User
Error-Text	Textuelle Erklärung	Alle User
aria-invalid="true"	Programmatischer Hinweis	Screenreader-User

Alle vier Indikatoren zusammen = WCAG 2.1 AA konform

## Success States

### Visueller Indikator

Erfolgreich validierte Felder können optional mit einem Success-State gekennzeichnet werden:

1. **Gruene Border:** {input.field.border.success} (color.success)
2. **Success-Icon:** Check (Lucide) im Input rechts (16px, success color)
3. **Kein zusätzlicher Text notig** (visuelle Bestätigung reicht)

```
<div class="form-field form-field--success">
  <label for="email">E-Mail-Adresse</label>
  <div class="input-wrapper">
    <input
      type="email"
      id="email"
      value="max@firma.de"
      class="input--success"
    />
    <svg class="input-icon input-icon--success" aria-hidden="true">
      <!-- Check Icon -->
    </svg>
  </div>
</div>
```

## Verwendung

### Wann Success-State verwenden?

- Nach erfolgreicher Validierung komplexer Felder (z.B. E-Mail-Format, Passwort-Stärke)
- Bei asynchroner Validierung (z.B. "Benutzername verfügbar")
- Um User zu bestätigen dass Eingabe korrekt ist

### Wann NICHT verwenden?

- Bei jedem Feld (überladen, visueller Noise)
- Bei einfachen Feldern (Name, Stadt - offensichtlich korrekt)
- Wenn User offensichtlich richtig eingegeben hat

## Token-Werte

```
.input--success {
  border-color: var(--input-field-border-success); /* color.success */
}

.input-icon--success {
  color: var(--input-success-color); /* color.success */
  width: 16px;
  height: 16px;
}
```

## Fehlermeldungen

### Stil: Erklärend-Hilfsbereit

Fehlermeldungen müssen NICHT nur sagen "was ist falsch", sondern auch "wie kann es korrigiert werden".

#### Struktur einer guten Fehlermeldung:

1. **Was ist das Problem?** (z.B. "Die E-Mail-Adresse ist ungültig")
2. **Wie beheben?** (z.B. "Bitte prüfe das Format")
3. **Beispiel (optional):** (z.B. "z.B. `name@firma.de`")

## Beispiele

Schlecht	Gut
"Ungueltig"	"Bitte gib eine gueltige E-Mail-Adresse ein (z.B. <a href="mailto:name@firma.de">name@firma.de</a> )"
"Fehler"	"Das Passwort muss mindestens 8 Zeichen lang sein"
"Falsches Format"	"Bitte gib das Datum im Format TT.MM.JJJJ ein (z.B. 01.12.2024)"
"Pflichtfeld"	"Bitte gib deinen Namen ein"
"Die Eingabe ist zu kurz"	"Der Name muss mindestens 2 Zeichen haben"
"Nicht erlaubt"	"Das Passwort darf keine Leerzeichen enthalten"
"Fehlerhafte Eingabe"	"Die Telefonnummer muss mit + und Laendervorwahl beginnen (z.B. +49 123 456789)"

## Ton

Fehlermeldungen sollten:

- **Hoeflich sein:** "Bitte gib..." statt "Du hast vergessen..."
- **Nicht anklagend sein:** "Die E-Mail-Adresse ist ungültig" statt "Du hast eine ungültige E-Mail eingegeben"
- **Aktive Sprache verwenden:** "Bitte gib..." statt "Es wurde nicht..."
- **Konkret sein:** "Mindestens 8 Zeichen" statt "Zu kurz"
- **Hilfreich sein:** Beispielwerte angeben wo sinnvoll

## Zod Schema Beispiele

```

import { z } from 'zod';

const schema = z.object({
  name: z
    .string()
    .min(1, 'Bitte gib deinen Namen ein')
    .min(2, 'Der Name muss mindestens 2 Zeichen haben'),

  email: z
    .string()
    .min(1, 'E-Mail-Adresse ist erforderlich')
    .email('Bitte gib eine gueltige E-Mail-Adresse ein (z.B. name@firma.de)',

  phone: z
    .string()
    .regex(
      `/^\+[0-9]{1,3}\s[0-9\s]+$/,
      'Bitte gib die Telefonnummer mit Laendervorwahl ein (z.B. +49 123 456789)'
    ),
    .max(15, 'Die Telefonnummer darf maximal 15 Zeichen haben'),

  password: z
    .string()
    .min(8, 'Das Passwort muss mindestens 8 Zeichen lang sein')
    .regex(/^[A-Z]/, 'Das Passwort muss mindestens einen Grossbuchstaben enthalten')
    .regex(/^[0-9]/, 'Das Passwort muss mindestens eine Zahl enthalten'),

  message: z
    .string()
    .min(1, 'Bitte gib eine Nachricht ein')
    .min(10, 'Die Nachricht muss mindestens 10 Zeichen haben')
    .max(500, 'Die Nachricht darf maximal 500 Zeichen haben'),

  birthdate: z
    .string()
    .regex(/^\d{2}.\d{2}.\d{4}$/, 'Bitte gib das Datum im Format TT.MM.JJJJ ein (z.B. 01.12.1990)')
  });
}

```

## Error-Position

### Inline (Standard)

Fehlermeldungen stehen direkt UNTER dem fehlerhaften Feld:

```

[Label]
[Input Field] ← Roter Border
[Error Message] ← Direkt darunter

```

Vorteile:

- Sofort sichtbar im Kontext
- User sieht Fehler genau dort wo er ist

- Keine Navigation noetig

**Empfohlen fuer:**

- Einfache Formulare (1-5 Felder)
- Single-Page-Formulare
- Standard-Kontaktformulare

### Inline + Summary (Komplexe Formulare)

Bei komplexen Formularen (5+ Felder) zusaetzlich eine Error-Summary oben:

[Error Summary: 3 Fehler gefunden]

- E-Mail-Adresse ungultig
- Passwort zu kurz
- AGB nicht akzeptiert

[Formularfelder mit inline Errors]

**Vorteile:**

- Ueberblick ueber alle Fehler
- Links zu Feldern (schnelle Navigation)
- Focus springt zur Summary bei Submit

**Empfohlen fuer:**

- Formulare mit 5+ Feldern
- Multi-Step-Formulare
- Checkout-Prozesse

### Summary-Pattern

```
<div class="error-summary" role="alert" tabindex="-1" id="error-summary">
  <h2 class="error-summary__title">Bitte korrigiere folgende Fehler:</h2>
  <ul class="error-summary__list">
    <li>
      <a href="#email">E-Mail-Adresse ist ungultig</a>
    </li>
    <li>
      <a href="#password">Passwort zu kurz</a>
    </li>
    <li>
      <a href="#terms">Bitte akzeptiere die Nutzungsbedingungen</a>
    </li>
  </ul>
</div>
```

**Focus-Management:**

```

const onSubmit = (data) => {
  // Validierung schlaegt fehl
  if (errors) {
    // Focus auf Error-Summary setzen
    document.getElementById('error-summary')?.focus();
  }
};

```

Styling:

```

.error-summary {
  background-color: #fef2f2;           /* Helles Rot */
  border: 2px solid var(--color-error); /* Rote Border */
  border-radius: 4px;
  padding: 16px;
  margin-bottom: 24px;
}

.error-summary__title {
  color: var(--color-error);
  font-size: 16px;
  font-weight: 600;
  margin: 0 0 8px 0;
}

.error-summary__list {
  margin: 0;
  padding-left: 20px;
}

.error-summary__list a {
  color: var(--color-error);
  text-decoration: underline;
}

.error-summary__list a:hover {
  text-decoration: none;
}

```

## Accessibility

### WCAG 3.3.1: Fehlererkennung

**Anforderung:** Wenn ein Eingabefehler automatisch erkannt wird, wird das fehlerhafte Element identifiziert und der Fehler wird dem Benutzer in Textform beschrieben.

**Umsetzung:**

- Fehler werden automatisch erkannt (Progressive Validation)
- Fehlerhafte Felder werden visuell identifiziert (rote Border + Icon)
- Fehler werden programmatisch identifiziert (aria-invalid="true")
- Fehler werden in Textform beschrieben (Error-Message)

### WCAG 3.3.3: Fehlervorschlaege

**Anforderung:** Wenn ein Eingabefehler automatisch erkannt wird und Korrekturvorschlaege bekannt sind, werden diese dem Benutzer mitgeteilt, es sei denn, dies wuerde die Sicherheit oder den Zweck des Inhalts gefaehrden.

#### Umsetzung:

- Fehlermeldungen enthalten Korrekturvorschlaege
- Beispielwerte werden angegeben wo hilfreich
- Format-Anforderungen werden erklaert

### WCAG 3.3.4: Fehlervermeidung (Rechtliches, Finanzielles, Daten)

**Anforderung:** Fuer Web-Seiten, die rechtliche Verpflichtungen oder finanzielle Transaktionen mit dem Nutzer zur Folge haben, ist mindestens eine der folgenden Aussagen wahr:

1. **Rueckgaengig:** Uebertragungen sind rueckgaengig zu machen
2. **Ueberprueft:** Vom Benutzer eingegebene Daten werden auf Eingabefehler ueberprueft und der Benutzer erhaelt die Gelegenheit, diese zu korrigieren
3. **Bestaetigt:** Es gibt einen Mechanismus zur Ueberpruefung, Bestaetigung und Korrektur von Informationen, bevor die Uebertragung abgeschlossen wird

#### Umsetzung fuer kritische Formulare:

```
<!-- Step 1: Dateneingabe mit Validierung -->
<form onsubmit="goToReview()">
    <!-- Formularfelder mit Progressive Validation -->
</form>

<!-- Step 2: Review & Bestaetigung -->
<div class="review-section">
    <h2>Bitte ueberpruefe deine Angaben</h2>

    <dl>
        <dt>Name:</dt>
        <dd>Max Mustermann</dd>

        <dt>E-Mail:</dt>
        <dd>max@firma.de</dd>
    </dl>

    <button onclick="goBack()">Zurueck zur Bearbeitung</button>
    <button onclick="submitFinal()">Jetzt kostenpflichtig bestellen</button>
</div>
```

#### aria-invalid

`aria-invalid="true"` kennzeichnet fehlerhafte Felder fuer Screenreader:

```

<input
  type="email"
  id="email"
  aria-invalid="true"
  aria-describedby="email-error"
/>

  Bitte gib eine gueltige E-Mail-Adresse ein


```

#### Werte:

- aria-invalid="false" - Feld ist gültig (Standard, kann weggelassen werden)
- aria-invalid="true" - Feld ist ungültig
- aria-invalid="grammar" - Grammatikfehler (selten verwendet)
- aria-invalid="spelling" - Rechtschreibfehler (selten verwendet)

#### role="alert"

Error-Messages benötigen `role="alert"`:

```

<span id="email-error" class="error-message" role="alert">
  Bitte gib eine gueltige E-Mail-Adresse ein
</span>

```

#### Warum wichtig?

- `role="alert"` wird sofort von Screenreadern angekündigt
- Impliziert `aria-live="assertive"` (unterbricht aktuelle Ausgabe)
- User bekommt sofortiges Feedback

## Focus-Management bei Submit

Wenn Submit fehlschlägt:

- Bei **Inline-Errors**: Focus auf erstes fehlerhaftes Feld
- Bei **Error-Summary**: Focus auf Summary

```

const onSubmit = async (data) => {
  // Validierung
  if (errors) {
    // Option 1: Focus auf erstes fehlerhaftes Feld
    const firstErrorField = document.querySelector('[aria-invalid="true"]');
    firstErrorField?.focus();

    // Option 2: Focus auf Error-Summary (wenn vorhanden)
    document.getElementById('error-summary')?.focus();
  }
};

```

Error-Summary mit `tabindex="-1"`:

```
<div class="error-summary" role="alert" tabindex="-1" id="error-summary">
  <!-- ... -->
</div>
```

`tabindex="-1"` ermöglicht programmatischen Focus (JavaScript), aber nicht Tab-Navigation.

## Links in Summary führen zu Feldern

```
<ul class="error-summary__list">
  <li>
    <a href="#email" onclick="document.getElementById('email').focus()">
      E-Mail-Adresse ist ungültig
    </a>
  </li>
</ul>
```

## Formular-Level-Validierung

### Bei Submit

Wenn User auf Submit klickt:

1. Alle Felder validieren (auch wenn noch nicht touched)
2. Alle Fehler gleichzeitig anzeigen (nicht schrittweise)
3. Focus auf ersten Fehler oder Summary
4. Submit verhindern bis alle Fehler korrigiert

```
const handleSubmit = async (e) => {
  e.preventDefault();

  // Validierung triggern
  const isValid = await form.trigger();

  if (!isValid) {
    // Focus Management
    const firstError = Object.keys(form.formState.errors)[0];
    form.setFocus(firstError);
    return;
  }

  // Submit durchführen
  await submitData(form.getValues());
};
```

### Submit-Button-Zustand

NICHT deaktivieren basierend auf Validierung:

```
<!-- FALSCH -->
<button type="submit" disabled={!isValid}>
  Absenden
</button>
```

#### Warum nicht?

- User weiss nicht WARUM Button deaktiviert ist
- Keine Feedback, welche Felder fehlerhaft sind
- Frustrierend wenn Button nicht klickbar ist

#### RICHTIG: Immer klickbar lassen:

```
<!-- RICHTIG -->
<button type="submit">
  Absenden
</button>
```

Validierung erfolgt bei Klick, User bekommt klare Fehler-Messages.

#### AUSNAHME: Deaktivieren während Submission:

```
<button type="submit" disabled={isSubmitting}>
  {isSubmitting ? 'Wird gesendet...' : 'Absenden'}
</button>
```

Verhindert Doppel-Submission.

## Vollstaendiges Integrations-Beispiel

Das folgende Beispiel zeigt alle Teile zusammen: Token-Referenzen, React Hook Form, Zod Schema, ARIA-Attribute und visuelles Styling.

## Schema (Zod)

```

import { z } from 'zod';

const contactSchema = z.object({
  name: z
    .string()
    .min(1, 'Bitte gib deinen Namen ein')
    .min(2, 'Der Name muss mindestens 2 Zeichen haben'),

  email: z
    .string()
    .min(1, 'E-Mail-Adresse ist erforderlich')
    .email('Bitte gib eine gueltige E-Mail-Adresse ein (z.B. name@firma.de)'),

  phone: z
    .string()
    .optional()
    .refine(
      (val) => !val || /^[+]{1}[0-9]{1,3}\s[0-9\s]+$/ .test(val),
      'Bitte gib die Telefonnummer mit Laendervorwahl ein (z.B. +49 123 456789)'
    ),

  message: z
    .string()
    .min(1, 'Bitte gib eine Nachricht ein')
    .min(10, 'Die Nachricht muss mindestens 10 Zeichen haben')
    .max(500, 'Die Nachricht darf maximal 500 Zeichen haben'),

  terms: z
    .boolean()
    .refine((val) => val === true, 'Bitte akzeptiere die Nutzungsbedingungen')
});

type ContactFormData = z.infer<typeof contactSchema>;

```

## React Component

```

import { useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import { AlertCircle, Check } from 'lucide-react';

function ContactForm() {
  const {
    register,
    handleSubmit,
    formState: { errors, isSubmitting, isValid, touchedFields }
  } = useForm<ContactFormData>({
    resolver: zodResolver(contactSchema),
    mode: 'onBlur',           // Initiale Validierung bei blur
    reValidateMode: 'onChange' // Nach Fehler: live re-validieren
  });

  const onSubmit = async (data: ContactFormData) => {
    try {
      // API-Call
      await fetch('/api/contact', {
        method: 'POST',
        body: JSON.stringify(data)
      });

      // Success-Handling
      alert('Nachricht gesendet!');
    } catch (error) {
      // Error-Handling
      console.error('Fehler beim Senden:', error);
    }
  };

  // Helper: Zeige Success-State nur wenn touched + valid + kein Error
  const showSuccess = (fieldName: keyof ContactFormData) => {
    return touchedFields[fieldName] && !errors[fieldName];
  };
}

return (
  <form onSubmit={handleSubmit(onSubmit)} noValidate>
    {/* Error Summary (optional, bei komplexen Formularen) */}
    {Object.keys(errors).length > 0 && (
      <div className="error-summary" role="alert" tabIndex="-1">
        <h2 className="error-summary__title">
          Bitte korrigiere folgende Fehler:
        </h2>
        <ul className="error-summary__list">
          {errors.name && (
            <li>
              <a href="#name">{errors.name.message}</a>
            </li>
          )}
          {errors.email && (
            <li>
              <a href="#email">{errors.email.message}</a>
            </li>
          )}
        </ul>
      </div>
    )}
  </form>
);

```

```

{errors.phone && (
  <li>
    <a href="#phone">{errors.phone.message}</a>
  </li>
)
{errors.message && (
  <li>
    <a href="#message">{errors.message.message}</a>
  </li>
)
{errors.terms && (
  <li>
    <a href="#terms">{errors.terms.message}</a>
  </li>
)
</ul>
</div>
)}

/* Name Field */
<div className={`form-field ${errors.name ? 'form-field--error' : ''}`}>
  <label htmlFor="name" className={errors.name ? 'error' : ''}>
    Name
    <span className="required" aria-label="Pflichtfeld">*</span>
  </label>
  <div className="input-wrapper">
    <input
      id="name"
      type="text"
      className={errors.name ? 'input--error' : ''}
      aria-required="true"
      aria-invalid={!errors.name}
      aria-describedby={errors.name ? 'name-error' : undefined}
      {...register('name')}
    />
    {errors.name && (
      <AlertCircle className="input-icon input-icon--error" aria-hidden="true" />
    )}
  </div>
  {errors.name && (
    <span id="name-error" className="error-message" role="alert">
      {errors.name.message}
    </span>
  )}
</div>

/* Email Field with Success State */
<div className={`form-field ${errors.email ? 'form-field--error' : showSuccess('email') ? 'form-field--success' : ''}`}>
  <label htmlFor="email" className={errors.email ? 'error' : ''}>
    E-Mail-Adresse
    <span className="required" aria-label="Pflichtfeld">*</span>
  </label>
  <div className="input-wrapper">
    <input
      id="email"
      type="email"
      className={errors.email ? 'input--error' : showSuccess('email') ? 'input--success' : ''}
      aria-required="true"
      aria-invalid={!errors.email}
    />
  </div>
</div>

```

```

        aria-describedby={errors.email ? 'email-error' : 'email-helper'}
        {...register('email')}
    />
{errors.email && (
    <AlertCircle className="input-icon input-icon--error" aria-hidden="true" />
)}
{showSuccess('email') && (
    <Check className="input-icon input-icon--success" aria-hidden="true" />
)}
</div>
{!errors.email && (
    <span id="email-helper" className="helper-text">
        Wir verwenden deine E-Mail nur zur Kontaktaufnahme
    </span>
)}
{errors.email && (
    <span id="email-error" className="error-message" role="alert">
        {errors.email.message}
    </span>
)}
</div>

/* Phone Field (Optional) */
<div className={`${form-field ${errors.phone ? 'form-field--error' : ''}`}>
    <label htmlFor="phone">
        Telefonnummer
        <span className="optional">(optional)</span>
    </label>
    <div className="input-wrapper">
        <input
            id="phone"
            type="tel"
            placeholder="+49 123 456789"
            className={errors.phone ? 'input--error' : ''}
            aria-invalid={!errors.phone}
            aria-describedby={errors.phone ? 'phone-error' : 'phone-helper'}
            {...register('phone')}
        />
{errors.phone && (
    <AlertCircle className="input-icon input-icon--error" aria-hidden="true" />
)}
</div>
{!errors.phone && (
    <span id="phone-helper" className="helper-text">
        Bitte mit Laendervorwahl
    </span>
)}
{errors.phone && (
    <span id="phone-error" className="error-message" role="alert">
        {errors.phone.message}
    </span>
)}
</div>

/* Message Field (Textarea) */
<div className={`${form-field ${errors.message ? 'form-field--error' : ''}`}>
    <label htmlFor="message" className={errors.message ? 'error' : ''}>
        Nachricht
        <span className="required" aria-label="Pflichtfeld">*</span>
    </label>

```

```

</label>
<textarea
  id="message"
  rows={5}
  className={errors.message ? 'input--error' : ''}
  aria-required="true"
  aria-invalid={!errors.message}
  aria-describedby={errors.message ? 'message-error' : 'message-helper'}
  {...register('message')}
/>
{!errors.message && (
  <span id="message-helper" className="helper-text">
    Mindestens 10 Zeichen, maximal 500 Zeichen
  </span>
)}
{errors.message && (
  <span id="message-error" className="error-message" role="alert">
    {errors.message.message}
  </span>
)}
</div>

/* Terms Checkbox */
<div className={`form-field ${errors.terms ? 'form-field--error' : ''}`}>
  <label className="checkbox-label">
    <input
      type="checkbox"
      id="terms"
      className="checkbox-native"
      aria-required="true"
      aria-invalid={!errors.terms}
      {...register('terms')}
    />
    <span className="checkbox-custom"></span>
    Ich akzeptiere die{' '}
    <a href="/terms" target="_blank">Nutzungsbedingungen</a>
    <span className="required" aria-label="Pflichtfeld">*</span>
  </label>
  {errors.terms && (
    <span className="error-message" role="alert">
      {errors.terms.message}
    </span>
  )}
</div>

/* Submit Button */
<button type="submit" disabled={isSubmitting} className="button button--primary">
  {isSubmitting ? 'Wird gesendet...' : 'Absenden'}
</button>
</form>
);
}

export default ContactForm;

```

## CSS mit Token-Variablen

```

/* Form Field Container */
.form-field {
  margin-bottom: var(--form-spacing-field-to-field); /* 16px */
}

.form-field--error label {
  color: var(--input-label-color-error); /* color.error */
}

/* Labels */
.form-field label {
  display: block;
  font-size: var(--input-label-font-size); /* 14px */
  font-weight: var(--input-label-font-weight); /* 500 */
  color: var(--input-label-color-default); /* neutral.700 */
  margin-bottom: var(--input-label-margin-bottom); /* 8px */
}

.required {
  color: var(--input-required-color); /* error */
  margin-left: 2px;
}

.optional {
  color: var(--neutral-500);
  font-weight: 400;
  margin-left: 4px;
}

/* Input Wrapper (for icon positioning) */
.input-wrapper {
  position: relative;
  display: flex;
  align-items: center;
}

/* Input Fields */
.form-field input[type="text"],
.form-field input[type="email"],
.form-field input[type="tel"],
.form-field textarea {
  width: 100%;
  height: var(--input-height-md); /* 40px */
  padding: 0 var(--input-padding-x-md); /* 12px */
  border: var(--input-field-border-width-default) solid var(--input-field-border-default);
  border-radius: var(--input-base-border-radius); /* 4px */
  font-size: var(--input-font-size-md); /* 16px */
  font-family: var(--input-base-font-family); /* Inter */
  color: var(--input-field-text-default); /* neutral.900 */
  background-color: var(--input-field-background-default); /* white */
  transition: var(--input-base-transition); /* 150ms ease-in-out */
}

.form-field textarea {
  height: auto;
}

```

```
min-height: var(--input-textarea-min-height); /* 120px */
padding: var(--input-padding-x-md);
resize: var(--input-textarea-resize); /* vertical */
}

/* Input with icon - add padding for icon space */
.input-wrapper input {
  padding-right: 40px;
}

/* Input States: Hover */
.form-field input:hover,
.form-field textarea:hover {
  border-color: var(--input-field-border-hover); /* neutral.400 */
}

/* Input States: Focus */
.form-field input:focus-visible,
.form-field textarea:focus-visible {
  border-color: var(--input-field-border-focus); /* hydrophon.blau.500 */
  border-width: var(--input-field-border-width-focus); /* 2px */
  outline: var(--input-focus-ring-width) solid var(--input-focus-ring-color);
  outline-offset: var(--input-focus-outline-offset); /* 0px */
}

/* Input States: Error */
.input--error {
  border-color: var(--input-field-border-error) !important; /* error */
}

/* Input States: Success */
.input--success {
  border-color: var(--input-field-border-success) !important; /* success */
}

/* Input States: Disabled */
.form-field input:disabled,
.form-field textarea:disabled {
  background-color: var(--input-field-background-disabled); /* neutral.50 */
  border-color: var(--input-field-border-disabled); /* neutral.200 */
  color: var(--input-field-text-disabled); /* neutral.600 */
  cursor: not-allowed;
}

/* Input Icons */
.input-icon {
  position: absolute;
  right: 12px;
  width: 16px;
  height: 16px;
  pointer-events: none;
}

.input-icon--error {
  color: var(--input-error-color); /* error */
}

.input-icon--success {
  color: var(--input-success-color); /* success */
}
```

```
}

/* Helper Text */
.helper-text {
  display: block;
  font-size: var(--input-helper-font-size);      /* 12px */
  color: var(--input-helper-color);                /* neutral.600 */
  margin-top: var(--input-helper-margin-top);      /* 4px */
  line-height: 1.4;
}

/* Error Message */
.error-message {
  display: block;
  font-size: var(--input-error-font-size);        /* 12px */
  color: var(--input-error-color);                /* error */
  margin-top: var(--input-error-margin-top);        /* 4px */
  line-height: 1.4;
}

/* Error Summary */
.error-summary {
  background-color: #fef2f2;
  border: 2px solid var(--color-error);
  border-radius: 4px;
  padding: 16px;
  margin-bottom: 24px;
}

.error-summary__title {
  color: var(--color-error);
  font-size: 16px;
  font-weight: 600;
  margin: 0 0 8px 0;
}

.error-summary__list {
  margin: 0;
  padding-left: 20px;
}

.error-summary__list a {
  color: var(--color-error);
  text-decoration: underline;
}

.error-summary__list a:hover {
  text-decoration: none;
}

/* Checkbox Styling (Native-First) */
.checkbox-label {
  display: flex;
  align-items: flex-start;
  gap: var(--checkbox-label-gap);                  /* 8px */
  cursor: pointer;
}

.checkbox-native {
```

```
position: absolute;
opacity: 0;
width: var(--checkbox-size-default); /* 20px */
height: var(--checkbox-size-default); /* 20px */
}

.checkbox-custom {
  display: inline-block;
  width: var(--checkbox-size-default); /* 20px */
  height: var(--checkbox-size-default); /* 20px */
  border: var(--checkbox-border-width) solid var(--checkbox-border-default);
  border-radius: var(--checkbox-border-radius); /* 4px */
  background-color: var(--checkbox-background-default);
  position: relative;
  flex-shrink: 0;
  margin-top: 2px; /* Optical alignment with text */
}

.checkbox-native:checked + .checkbox-custom {
  background-color: var(--checkbox-background-checked);
  border-color: var(--checkbox-border-checked);
}

.checkbox-native:checked + .checkbox-custom::after {
  content: '';
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: var(--checkbox-icon-size); /* 14px */
  height: var(--checkbox-icon-size);
  /* Check icon as background-image or inline SVG */
  background-image: url("data:image/svg+xml,...");
}

/* Submit Button */
.button {
  height: var(--input-height-md); /* 40px - aligned with inputs */
  padding: 0 var(--button-padding-x-md);
  border-radius: var(--input-base-border-radius); /* 4px */
  font-size: var(--input-font-size-md); /* 16px */
  font-weight: 500;
  cursor: pointer;
  transition: var(--input-base-transition);
}

.button--primary {
  background-color: var(--color-action-primary-default);
  color: white;
  border: none;
}

.button--primary:hover:not(:disabled) {
  background-color: var(--color-action-primary-hover);
}

.button--primary:disabled {
  background-color: var(--neutral-300);
  cursor: not-allowed;
```

```
    opacity: 0.6;
}
```

## Design Tokens

Alle Design Tokens fuer Validierung und Error States:

Token	Wert	Verwendung
input.field.border.error	color.error	Input Border im Error-State
input.field.border.success	color.success	Input Border im Success-State
input.field.background.error	neutral.white	Input Hintergrund Error (bleibt weiss)
input.field.background.success	neutral.white	Input Hintergrund Success (bleibt weiss)
input.error.color	color.error	Error Message Textfarbe
input.error.fontSize	12px	Error Message Schriftgroesse
input.error.marginTop	4px	Abstand Input → Error Message
input.success.color	color.success	Success Icon/Text Farbe
input.label.color.error	color.error	Label Farbe im Error-State

## Anti-Patterns

### Vermeiden

#### 1. Fehlermeldungen erst bei Submit (zu spaet)

```
// FALSCH
const form = useForm({
  mode: 'onSubmit' // User sieht Fehler erst am Ende
});
```

**Warum?** Bei komplexen Formularen sieht User alle Fehler auf einmal, muss alles korrigieren, hohe Abbruchrate.

#### 2. Validierung bei jedem Tastendruck von Anfang an (zu frueh)

```
// FALSCH
const form = useForm({
  mode: 'onChange' // "E-Mail ungultig" nach dem ersten Buchstaben
});
```

**Warum?** Frustrierend, User kann nicht in Ruhe tippen, lenkt von Eingabe ab.

#### 3. Generische Meldungen wie "Fehler" oder "Ungultig"

```
<!-- FALSCH -->
Fehler</span>

<!-- RICHTIG -->

  Bitte gib eine gueltige E-Mail-Adresse ein (z.B. name@firma.de)
</span>
```

**Warum?** User weiss nicht, WAS falsch ist oder WIE es zu korrigieren ist.

#### 4. Submit-Button deaktivieren statt Feedback geben

```
<!-- FALSCH -->
<button type="submit" disabled={!isValid}>
  Absenden
</button>
```

**Warum?** User weiss nicht, warum Button deaktiviert ist, keine Hilfestellung.

#### 5. Nur rote Farbe als Fehlerindikator

```
<!-- FALSCH - nur Farbe -->
<input style="border-color: red;" />

<!-- RICHTIG - Farbe + Icon + Text -->
<input class="input--error" aria-invalid="true" aria-describedby="error" />
<AlertCircle aria-hidden="true" />
<span id="error" role="alert">Fehlermeldung</span>
```

**Warum?** WCAG 1.4.1 - Farbenblinde User sehen keinen Unterschied.

#### 6. Error-Meldungen die verschwinden nach kurzer Zeit

```
// FALSCH
setTimeout(() => {
  setErrorMessage('');
}, 3000); // Fehler verschwindet nach 3 Sekunden
```

**Warum?** User hat vielleicht nicht genug Zeit zum Lesen, Screenreader-User verpasst die Meldung komplett.

#### Siehe auch:

- [Labels und Helper Text](#)
- [Text Input Dokumentation](#)
- [Checkbox Dokumentation](#)
- [Radio Button Dokumentation](#)

# Navigation & Content Structure

---

## Übersicht

Phase 4 des Hydrophon Design-Systems umfasst Navigation-Komponenten für die Website-Struktur sowie Content-Darstellungskomponenten für Produkte und Inhalte.

**Ziel:** Nutzer können sich effizient durch die Website navigieren, ihre Position in der Hierarchie verstehen und Inhalte schnell erfassen.

**Komponenten:** Navigation (Header, Mobile Drawer, Breadcrumb, Footer) und Content-Struktur (Cards, Tables)

## Navigation-Komponenten

### Header Navigation

**Zweck:** Primäre Website-Navigation mit Logo und Hauptmenü

**Dokumentation:** [header.md](#)

**Eigenschaften:**

- Logo links + Navigation rechts (F-Pattern Reading Flow)
- Horizontale Menüstruktur für Desktop
- Sticky Behavior auf Desktop (Mobile: statisch)
- WCAG AAA Touch Targets (44px)
- `aria-current="page"` für aktiven Link
- Server-Side Rendering empfohlen (keine FOIS)

**Tokens:** `navigation.header.*` (48+ Tokens)

**Verwendung:** Jede Website-Seite

---

## Mobile Drawer

**Zweck:** Slide-Out Navigation für mobile Geräte

**Dokumentation:** [mobile-drawer.md](#)

**Eigenschaften:**

- Hamburger-Toggle mit X-Icon für Close
- Links-seitiger Slide-Out mit Overlay
- Focus-Trap Pattern (Tab wrapping)
- ESC-Key schließt Drawer (WCAG 2.1.2)
- Focus auf Close-Button beim Öffnen
- Transform-basierte Animation (GPU-beschleunigt)
- Body-Scroll Lock während Drawer offen

**Tokens:** `navigation.drawer.*`

**Verwendung:** Mobile Viewport (< 768px)

## Breadcrumb Navigation

**Zweck:** Hierarchische Position innerhalb der Website anzeigen

**Dokumentation:** [breadcrumb.md](#)

**Eigenschaften:**

- Chevron (>) als Separator
- `<ol>` für semantische Liste
- `aria-current="page"` für aktuelle Seite
- Automatisches Screen Reader Hiding für Separatoren (CSS ::after)
- Responsive: Collapsing auf Mobile (nur Parent + Current)

**Tokens:** `navigation.breadcrumb.*`

**Verwendung:** Unterseiten mit Hierarchie (nicht Homepage)

**Beispiel-Pfad:**

Home > Produkte > Ablaufrinnen > Hydrophon 80cm

---

## Footer Navigation

**Zweck:** Sekundäre Navigation, rechtliche Links, Kontaktinformationen

**Dokumentation:** [footer.md](#)

**Eigenschaften:**

- CSS Grid Layout (responsive Spalten)
- Mehrere `<nav>` mit eindeutigen `aria-label`
- Uppercase Headings mit Letter-Spacing
- `line-height: 2` für Touch-Target-Optimierung (28px bei 14px Font)
- Automatische `role="contentinfo"` durch `<footer>`-Element

**Tokens:** `navigation.footer.*`

**Verwendung:** Jede Website-Seite

**Typische Sektionen:**

- Produkte
- Unternehmen
- Service & Support
- Rechtliches (Impressum, Datenschutz, AGB)

---

## Content-Komponenten

### Product Card

**Zweck:** Produkte in Grid-Listen anzeigen

**Dokumentation:** [../components/product-card.md](#)

**Eigenschaften:**

- Quadratisches Bild (1:1 Aspect Ratio)
- Produktnamen + 2-3 Specs + CTA
- CSS Grid auto-fit Pattern (keine Media Queries)
- Hover-Effekt: Lift + Shadow (GPU-beschleunigt)
- Lazy Loading für Performance
- `<article>` für Semantik

**Tokens:** `card.*` (48+ Tokens)

**Verwendung:** Produktkataloge, Kategorie-Seiten, Suchergebnisse

**Grid-Pattern:**

```
grid-template-columns: repeat(auto-fit, minmax(min(280px, 100%), 1fr));
```

## Content Card

**Zweck:** Allgemeine Inhalte (Blog, News, Features) präsentieren

**Dokumentation:** [../components/content-card.md](#)

**Eigenschaften:**

- Flexibles Bildformat (16:9, 4:3, oder kein Bild)
- Meta-Informationen (Datum, Kategorie)
- Description mit `line-clamp` (max 3 Zeilen)
- Varianten: Vertikal, Horizontal, Text-Only, Feature mit Icon
- Gleiche Hover-Effekte wie Product Card

**Tokens:** `card.*` (shared mit Product Card)

**Verwendung:** Blog-Übersichten, News-Bereiche, Feature-Highlights

**Varianten:**

- **Vertikal:** Bild oben, Content unten
- **Horizontal:** Bild links, Content rechts
- **Text-Only:** Nur Content, kein Bild
- **Feature:** Icon statt Bild

## Data Table

**Zweck:** Strukturierte Daten in tabellarischer Form (Produktvergleiche)

**Dokumentation:** [../components/data-table.md](#)

**Eigenschaften:**

- Semantisches `<table>` mit `<caption>`
- `scope="col"` auf allen `<th>` in `<thead>`
- `scope="row"` auf ersten `<th>` in `<tbody>`
- Zebra-Striping: Subtiler Kontrast (neutral.50)
- Row Hover: Hydrophon-Blau.50
- Responsive: Horizontaler Scroll (empfohlen)

**Tokens:** `table.*` (36+ Tokens)

**Verwendung:** Produktvergleiche, technische Datenblätter, Preislisten

**Accessibility:**

- WCAG 1.3.1 - Info and Relationships
  - Screen Reader Table Commands
  - Native Keyboard Navigation
- 

## Token-Dateien

### **tokens/navigation.json**

**Inhalt:** Tokens für Header, Drawer, Breadcrumb, Footer

**Tokens:** 72+ Tokens

**Bereiche:**

- `navigation.header.*` – Header Heights, Link States, Logo Size
- `navigation.drawer.*` – Drawer Width, Overlay, Transition
- `navigation.breadcrumb.*` – Font Size, Colors, Separator
- `navigation.footer.*` – Grid, Link States, Headings

**Dokumentation:** [04-01-SUMMARY.md](#), [04-02-SUMMARY.md](#)

---

### **tokens/cards.json**

**Inhalt:** Tokens für Product Cards und Content Cards

**Tokens:** 48+ Tokens

**Bereiche:**

- `card.*` – Container, Padding, Border, Radius
- `card.shadow.*` – Default, Hover, Active
- `card.image.*` – Aspect Ratio, Background
- `card.title.*` – Font Size, Weight, Color
- `card.specs.*` – Font Size, Color, Line Height
- `card.cta.*` – Colors, Weight

**Dokumentation:** Siehe [Product Card](#) und [Content Card](#)

---

### **tokens/tables.json**

**Inhalt:** Tokens für Data Tables

**Tokens:** 36+ Tokens

**Bereiche:**

- `table.*` – Font Size, Background, Border
- `table.cell.*` – Padding
- `table.header.*` – Background, Color, Weight, Border
- `table.row.*` – Default, Stripe, Hover, Border
- `table.caption.*` – Font Size, Weight, Padding

- `table.responsive.*` – Min Width, Overflow

Dokumentation: Siehe [Data Table](#)

---

## Architektur-Entscheidungen

### Navigation

1. **Logo links + Navigation rechts:** Folgt B2B F-Pattern Reading Flow
2. **Desktop-only Sticky Header:** Mobile Viewport zu wertvoll (64px = 10% Screen)
3. **Hamburger-only Mobile:** Kein Desktop Hamburger, volle horizontale Navigation sichtbar
4. **44px Touch Targets:** WCAG AAA (nicht nur AA) für B2B-Outdoor-Umgebungen
5. **Transform-basierte Slide-Animation:** GPU-beschleunigt (translateX vs position)
6. **Server-Side aria-current:** Verhindert Flash of Incorrect State (FOIS)

### Content

1. **CSS Grid auto-fit:** Responsive ohne Media Queries (intrinsic design)
  2. **GPU-beschleunigter Hover:** `transform: translateY()` statt `top` oder `margin`
  3. **Shadow-Animation via ::after:** `opacity` statt `box-shadow` (Performance)
  4. **Horizontaler Scroll für Tables:** Behält Semantik für Screen Reader
  5. **Subtile Zebra-Stripes:** `neutral.50` (nicht stärker) für Lesbarkeit ohne visuelle Störung
- 

## Accessibility-Patterns

### ARIA-Patterns

- `aria-current="page"` – Aktiver Navigations-Link
- `aria-expanded` – Drawer-Toggle-State
- `aria-controls` – Verknüpfung Toggle → Drawer
- `aria-label` – Eindeutige Labels für Navigation-Landmarks
- `aria-sort` – Sortierungs-State für Tabellen (optional)

### Semantic HTML

- `<nav>` – Navigation-Bereiche
- `<header>` – Site-Header (automatisch `role="banner"`)
- `<footer>` – Site-Footer (automatisch `role="contentinfo"`)
- `<article>` – Cards (self-contained content)
- `<table>` – Tabellarische Daten (nicht divs)
- `<caption>` – Tabellen-Beschreibung

### Keyboard-Navigation

- **Tab:** Navigiert durch interaktive Elemente
- **Enter/Space:** Aktiviert Links/Buttons
- **ESC:** Schließt Drawer

- Arrow Keys: Screen Reader Table Navigation

## Focus Management

- `:focus-visible` – Keyboard-only Focus Indicators
  - Focus-Trap in Drawer – Tab wrapping innerhalb Modal
  - Focus auf Close-Button beim Drawer-Öffnen
- 

## Performance-Patterns

### CSS

- Transform statt Position: GPU-beschleunigt
- Pseudo-Element für Shadow-Animation: Nur `opacity` animieren
- will-change sparsam: Nur auf Hover/Focus-Kandidaten

### Images

- Lazy Loading: `loading="lazy"` auf alle Product/Content Images
- Async Decoding: `decoding="async"` für non-blocking
- Aspect Ratio: CSS `aspect-ratio` verhindert Layout Shift

### JavaScript

- Focus-Trap nur bei Open: Nicht pre-render
  - Body-Scroll Lock: `overflow: hidden` + padding-right kompensation
  - ResizeObserver für Responsive: Event-basiert statt Polling
- 

## Verwandte Phasen

### Phase 1: Foundation & Brand Identity

- Farben, Typografie, Spacing → Basis für alle Navigation/Content-Komponenten

### Phase 2: Icons & Basic Interactions

- Lucide Icons → Hamburger, Close, Chevrons in Navigation
- Button-Komponenten → CTAs in Cards

### Phase 3: Forms & Data Input

- Input-Patterns → Suchfeld in Header (falls implementiert)

## Weitere Ressourcen

### WCAG Guidelines

- [WCAG 2.1.2 - No Keyboard Trap](#)
- [WCAG 2.4.7 - Focus Visible](#)
- [WCAG 1.3.1 - Info and Relationships](#)

### ARIA Patterns

- [ARIA: navigation role](#)
- [ARIA: aria-current](#)
- [ARIA: Dialog \(Modal\) Pattern](#)

### CSS Patterns

- [CSS Grid auto-fit vs auto-fill](#)
- [CSS Tricks: Responsive Tables](#)
- [Web.dev: Lazy Loading Images](#)

### Performance

- [Google Web Fundamentals: Animations](#)
- [MDN: will-change](#)
- [Web.dev: prefers-reduced-motion](#)

---

**Phase 4 abgeschlossen:** Navigation & Content Structure vollständig dokumentiert mit 6 Komponenten (Header, Mobile Drawer, Breadcrumb, Footer, Product Card, Content Card, Data Table) und 156+ Tokens.

---

# Header Navigation (Desktop)

---

Die Header-Navigation ermöglicht Nutzer\*innen die primäre Orientierung auf der Website. Als persistentes Element bildet der Header den Einstieg in alle Hauptbereiche und kommuniziert die Markenidentität durch Logo-Platzierung.

**Anforderungen:** NAV-01 (Header-Struktur), NAV-02 (Navigation-Links), NAV-03 (Sticky-Behavior), NAV-04 (Responsive Breakpoints)

## Übersicht

Der Desktop-Header kombiniert Logo-Präsenz mit horizontaler Navigation. Die klare Aufteilung folgt dem F-Pattern-Lesefluss: Logo links (primäre visuelle Anker), Navigation rechts (Aktionselemente).

### Design-Prinzipien:

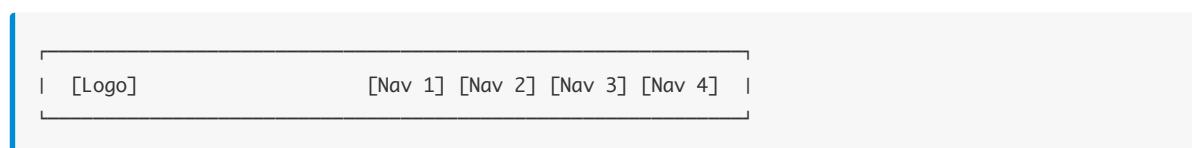
1. Klarheit — Maximal 7 Hauptnavigationspunkte für kognitive Verarbeitung
2. Konsistenz — Identische Header-Struktur über alle Seiten hinweg
3. Barrierefreiheit — Semantische HTML-Struktur mit ARIA-Landmarks
4. Performance — Leichtgewichtige Implementierung ohne JavaScript-Frameworks

### B2B-Kontext:

Professionelle B2B-Navigation vermeidet überladene Mega-Menüs und Dropdown-Strukturen. Der flache Header kommuniziert Übersichtlichkeit und Zielgerichtetetheit — wichtige Signale für Geschäftskunden mit begrenzter Zeit.

## Anatomie

Der Header besteht aus drei funktionalen Bereichen:



### 1. Logo-Bereich (Links)

- **Funktion:** Markenidentität, Link zur Startseite
- **Element:** `<a>` mit Logo-SVG
- **Sizing:** Minimum 120px Breite (aus Logo-Guidelines)
- **Verhalten:** Logo ist immer klickbar und führt zur Startseite

### 2. Hauptnavigation (Rechts)

- **Funktion:** Zugang zu allen Hauptbereichen
- **Element:** `<nav>` mit horizontaler Link-Liste
- **Layout:** Flexbox mit gap-spacing zwischen Links
- **Verhalten:** Hover-Feedback, aktiver Zustand für aktuelle Seite

### 3. Hamburger-Button (nur Mobile < 768px)

- **Funktion:** Öffnet Mobile-Drawer
  - **Dokumentation:** Siehe [mobile-drawer.md](#)
  - **Desktop:** `display: none` — vollständige Navigation ist sichtbar
- 

## Varianten

### Desktop (>= 768px)

Vollständige horizontale Navigation mit allen Links sichtbar.

#### Merkmale:

- Header-Höhe: 80px
- Horizontale Link-Anordnung
- Sticky-Behavior optional (siehe unten)
- Logo left + Nav right Layout

### Tablet/Mobile (< 768px)

Logo + Hamburger-Button, vollständige Navigation im Slide-Out Drawer.

#### Merkmale:

- Header-Höhe: 64px (kompakter für Viewport-Platzersparnis)
- Nur Logo und Hamburger sichtbar
- Navigation versteckt, wird über Drawer geöffnet
- Kein Sticky-Behavior (Mobile Viewport zu wertvoll)

#### Breakpoint:

```
@media (min-width: 768px) {
  /* Desktop: Vollständige Navigation */
  .header__nav {
    display: flex;
  }
  .header__toggle {
    display: none;
  }
}

@media (max-width: 767px) {
  /* Mobile: Nur Hamburger */
  .header__nav {
    display: none;
  }
  .header__toggle {
    display: flex;
  }
}
```

## Sticky-Behavior

Der Header kann optional als `position: sticky` implementiert werden, um beim Scrollen sichtbar zu bleiben.

**Empfehlung:** Nur auf Desktop aktivieren. Mobile Viewports profitieren von maximalem Content-Platz.

### Desktop Sticky mit Scroll-Shadow

Beim Scrollen erhält der Header einen subtilen Schatten für visuelle Trennung vom Content.

Implementierung:

```
<header class="header" id="main-header">
  <!-- Header Content -->
</header>
```

```
.header {
  position: sticky;
  top: 0;
  z-index: var(--navigation-header-z-index); /* 100 */
  background-color: var(--navigation-header-background);
  border-bottom: var(--navigation-header-border-bottom-width) solid var(--navigation-header-border);
  height: var(--navigation-header-height-desktop); /* 80px */
  transition: box-shadow 200ms ease-in-out;
}

.header--scrolled {
  box-shadow: var(--navigation-header-shadow); /* 0 1px 2px rgba(0,0,0,0.05) */
}

@media (max-width: 767px) {
  .header {
    position: relative; /* Kein Sticky auf Mobile */
    height: var(--navigation-header-height-mobile); /* 64px */
  }
}
```

JavaScript für Scroll-Detection:

```

const header = document.getElementById('main-header');
let lastScrollY = window.scrollY;

window.addEventListener('scroll', () => {
  const currentScrollY = window.scrollY;

  if (currentScrollY > 10) {
    header.classList.add('header--scrolled');
  } else {
    header.classList.remove('header--scrolled');
  }

  lastScrollY = currentScrollY;
});

```

**Performance-Tipp:** Verwende `requestAnimationFrame` oder Debouncing für bessere Scroll-Performance bei älteren Geräten.

## Navigation-Links

### Zustände

Navigation-Links durchlaufen vier visuelle Zustände:

#### 1. Default (Standard)

Der Ruhezustand für nicht-aktive Links.

**Tokens:**

- Farbe: `navigation.link.color.default` (neutral.700)
- Schriftgröße: `navigation.link.fontSize` (16px)
- Schriftgewicht: `navigation.link.fontWeight.default` (500 medium)

**CSS:**

```

.nav__link {
  color: var(--navigation-link-color-default);
  font-size: var(--navigation-link-font-size);
  font-weight: var(--navigation-link-font-weight-default);
  padding: var(--navigation-link-padding-y) var(--navigation-link-padding-x);
  text-decoration: none;
  transition: var(--navigation-link-transition);
}

```

#### 2. Hover (Maus darüber)

Visuelles Feedback bei Mouse-Over.

**Tokens:**

- Farbe: `navigation.link.color.hover` (neutral.900)

**CSS:**

```
.nav__link:hover {
  color: var(--navigation-link-color-hover);
}
```

### 3. Active (Aktuelle Seite)

Die aktuell angezeigte Seite erhält besondere Hervorhebung.

**Tokens:**

- Farbe: `navigation.link.color.active` (`hydrophon.blau.600`)
- Schriftgewicht: `navigation.link.fontWeight.active` (`600 semibold`)
- Indikator-Farbe: `navigation.activeIndicator.color` (`hydrophon.blau.500`)
- Indikator-Höhe: `navigation.activeIndicator.height` (`2px`)

**CSS:**

```
.nav__link[aria-current="page"] {
  color: var(--navigation-link-color-active);
  font-weight: var(--navigation-link-font-weight-active);
  position: relative;
}

.nav__link[aria-current="page"]::after {
  content: '';
  position: absolute;
  bottom: 0;
  left: var(--navigation-link-padding-x);
  right: var(--navigation-link-padding-x);
  height: var(--navigation-active-indicator-height);
  background-color: var(--navigation-active-indicator-color);
}
```

### 4. Focus (Tastatur-Navigation)

Sichtbarer Focus-Ring für Tastatur-Nutzer\*innen.

**Tokens:**

- Outline-Farbe: `navigation.focus.outline.color` (`hydrophon.blau.300`)
- Outline-Breite: `navigation.focus.outline.width` (`2px`)
- Outline-Offset: `navigation.focus.outline.offset` (`2px`)

**CSS:**

```
.nav__link:focus-visible {
  outline: var(--navigation-focus-outline-width) solid var(--navigation-focus-outline-color);
  outline-offset: var(--navigation-focus-outline-offset);
  border-radius: 2px;
}

/* Focus-Ring nur bei Tastatur-Navigation, nicht bei Maus-Klick */
.nav__link:focus:not(:focus-visible) {
  outline: none;
}
```

## aria-current Pattern

Das `aria-current="page"` Attribut kommuniziert die aktuelle Seite an assistive Technologien.

Regeln:

1. Nur EIN Element darf `aria-current="page"` haben
2. Server-seitig setzen — nicht per JavaScript nachladen (verhindert Flash of Incorrect State)
3. Nicht bei Logo — nur bei Navigation-Links
4. Kombination mit visuellem Indikator — Color + Underline + Font Weight

HTML-Beispiel:

```
<nav class="header__nav" aria-label="Hauptnavigation">
  <a href="/" class="nav__link">Home</a>
  <a href="/produkte" class="nav__link" aria-current="page">Produkte</a>
  <a href="/loesungen" class="nav__link">Lösungen</a>
  <a href="/support" class="nav__link">Support</a>
  <a href="/kontakt" class="nav__link">Kontakt</a>
</nav>
```

Server-seitig (Beispiel mit Template-Logik):

```
<a
  href="/produkte"
  class="nav__link"
  {{ if currentPage == "produkte" }}aria-current="page"{{ end }}
>
  Produkte
</a>
```

## Layout

### Flexbox-Container

```
.header {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  padding: 0 var(--navigation-header-padding-x-desktop); /* 24px */  
  height: var(--navigation-header-height-desktop); /* 80px */  
  max-width: var(--container-xl); /* 1280px */  
  margin: 0 auto;  
}
```

#### Layout-Prinzipien:

- `justify-content: space-between` — Logo links, Navigation rechts
- `align-items: center` — Vertikale Zentrierung beider Elemente
- `max-width: 1280px` — Begrenzt Header-Breite auf großen Bildschirmen
- `margin: 0 auto` — Zentriert Container horizontal

### Navigation-Liste

```
.header__nav {  
  display: flex;  
  gap: var(--navigation-link-gap); /* 24px */  
  align-items: center;  
  list-style: none;  
  margin: 0;  
  padding: 0;  
}
```

#### Spacing:

- `gap: 24px` zwischen Links (nicht margin/padding, um konsistente Abstände zu garantieren)
- Touch-Targets mindestens 44x44px (erfüllt durch padding + font-size)

## Tokens

Token-Name	Wert	Beschreibung
navigation.header.height.desktop	80px	Header-Höhe auf Desktop-Geräten
navigation.header.height.mobile	64px	Header-Höhe auf mobilen Geräten
navigation.header.background	#ffffff	Header Hintergrundfarbe (neutral.white)
navigation.header.borderBottom.width	1px	Untere Rahmenlinie Breite
navigation.header.borderBottom.color	#e5e5e5	Untere Rahmenlinie Farbe (neutral.200)
navigation.header.zIndex	100	Z-Index für Sticky-Verhalten
navigation.header.shadow	0 1px 2px ...	Schatten beim Scrollen (shadow.sm)
navigation.header.padding.x.desktop	24px	Horizontales Padding Desktop (spacing.6)
navigation.header.padding.x.mobile	16px	Horizontales Padding Mobile (spacing.4)
navigation.link.color.default	#404040	Link Standard-Farbe (neutral.700)
navigation.link.color.hover	#171717	Link Hover-Farbe (neutral.900)
navigation.link.color.active	#007bb8	Link aktiv-Farbe (hydrophon.blau.600)
navigation.link.fontSize	16px	Link Schriftgröße (fontSize.base)
navigation.link.fontWeight.default	500	Link Standard-Gewicht (fontWeight.medium)
navigation.link.fontWeight.active	600	Link aktiv-Gewicht (fontWeight.semibold)
navigation.link.padding.x	16px	Link horizontales Padding (spacing.4)
navigation.link.padding.y	12px	Link vertikales Padding (spacing.3)
navigation.link.gap	24px	Abstand zwischen Links (spacing.6)
navigation.activeIndicator.color	#008bd2	Aktiver Link Unterstreichung (hydrophon.blau.500)
navigation.activeIndicator.height	2px	Aktiver Link Unterstreichung Höhe
navigation.focus.outline.color	#5cc2f1	Focus Outline Farbe (hydrophon.blau.300)
navigation.focus.outline.width	2px	Focus Outline Breite
navigation.focus.outline.offset	2px	Focus Outline Abstand

### CSS-Variable-Mapping:

Tokens werden von Style Dictionary in CSS Custom Properties transformiert:

```

navigation.header.height.desktop → --navigation-header-height-desktop
navigation.link.color.default   → --navigation-link-color-default
navigation.focus.outline.color → --navigation-focus-outline-color

```

## Code-Beispiele

### Vollständiges HTML

```
<header class="header" role="banner">
  <div class="header__container">
    <!-- Logo-Bereich -->
    <a href="/" class="header__logo" aria-label="Zur Startseite">
      
    </a>

    <!-- Desktop Navigation -->
    <nav class="header__nav" aria-label="Hauptnavigation">
      <a href="/" class="nav__link">Home</a>
      <a href="/produkte" class="nav__link" aria-current="page">Produkte</a>
      <a href="/loesungen" class="nav__link">Lösungen</a>
      <a href="/support" class="nav__link">Support</a>
      <a href="/unternehmen" class="nav__link">Unternehmen</a>
      <a href="/kontakt" class="nav__link">Kontakt</a>
    </nav>

    <!-- Mobile Toggle (nur < 768px sichtbar) -->
    <button
      class="header__toggle"
      aria-label="Menü öffnen"
      aria-expanded="false"
      aria-controls="mobile-drawer"
    >
      <svg width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor">
        <line x1="3" y1="12" x2="21" y2="12" stroke-width="2"/>
        <line x1="3" y1="6" x2="21" y2="6" stroke-width="2"/>
        <line x1="3" y1="18" x2="21" y2="18" stroke-width="2"/>
      </svg>
    </button>
  </div>
</header>
```

## Vollständiges CSS

```
/* Header Container */
.header {
  position: sticky;
  top: 0;
  z-index: var(--navigation-header-z-index);
  background-color: var(--navigation-header-background);
  border-bottom: var(--navigation-header-border-bottom-width) solid var(--navigation-header-border);
  transition: box-shadow 200ms ease-in-out;
}

.header--scrolled {
  box-shadow: var(--navigation-header-shadow);
}

.header__container {
  display: flex;
  justify-content: space-between;
  align-items: center;
  height: var(--navigation-header-height-desktop);
  max-width: var(--container-xl);
  margin: 0 auto;
  padding: 0 var(--navigation-header-padding-x-desktop);
}

/* Logo */
.header__logo {
  display: flex;
  align-items: center;
  text-decoration: none;
}

.header__logo img {
  display: block;
  height: auto;
  max-height: 48px; /* Gibt Logo Raum innerhalb 80px Header */
}

/* Desktop Navigation */
.header__nav {
  display: flex;
  gap: var(--navigation-link-gap);
  align-items: center;
  margin: 0;
  padding: 0;
  list-style: none;
}

.nav__link {
  color: var(--navigation-link-color-default);
  font-size: var(--navigation-link-font-size);
  font-weight: var(--navigation-link-font-weight-default);
  padding: var(--navigation-link-padding-y) var(--navigation-link-padding-x);
  text-decoration: none;
  transition: var(--navigation-link-transition);
  position: relative;
}
```

```
}

.nav__link:hover {
  color: var(--navigation-link-color-hover);
}

.nav__link[aria-current="page"] {
  color: var(--navigation-link-color-active);
  font-weight: var(--navigation-link-font-weight-active);
}

.nav__link[aria-current="page"]::after {
  content: '';
  position: absolute;
  bottom: 0;
  left: var(--navigation-link-padding-x);
  right: var(--navigation-link-padding-x);
  height: var(--navigation-active-indicator-height);
  background-color: var(--navigation-active-indicator-color);
}

.nav__link:focus-visible {
  outline: var(--navigation-focus-outline-width) solid var(--navigation-focus-outline-color);
  outline-offset: var(--navigation-focus-outline-offset);
  border-radius: 2px;
}

/* Mobile Toggle (versteckt auf Desktop) */
.header__toggle {
  display: none;
}

/* Responsive: Mobile */
@media (max-width: 767px) {
  .header {
    position: relative; /* Kein Sticky */
  }

  .header__container {
    height: var(--navigation-header-height-mobile);
    padding: 0 var(--navigation-header-padding-x-mobile);
  }

  .header__nav {
    display: none; /* Versteckt auf Mobile, im Drawer sichtbar */
  }

  .header__toggle {
    display: flex;
    align-items: center;
    justify-content: center;
    width: var(--navigation-toggle-size);
    height: var(--navigation-toggle-size);
    padding: 0;
    border: none;
    background-color: var(--navigation-toggle-background-default);
    color: var(--navigation-toggle-color-default);
    border-radius: var(--navigation-toggle-border-radius);
    cursor: pointer;
  }
}
```

```

        transition: background-color 150ms ease-in-out, color 150ms ease-in-out;
    }

    .header__toggle:hover {
        background-color: var(--navigation-toggle-background-hover);
        color: var(--navigation-toggle-color-hover);
    }
}

```

## Barrierefreiheit

### Semantische HTML-Struktur

```

<header role="banner">
    <nav aria-label="Hauptnavigation">
        <!-- Navigation Links -->
    </nav>
</header>

```

#### ARIA-Landmarks:

- role="banner" auf <header> — Identifiziert primären Header (nur einmal pro Seite)
- <nav> — Semantisches Element für Navigationsbereich
- aria-label="Hauptnavigation" — Benennt Navigation für Screenreader

### Tastatur-Navigation

#### Anforderungen:

- Tab — Springt zum nächsten Link
- Shift + Tab — Springt zum vorherigen Link
- Enter — Aktiviert Link
- :focus-visible — Zeigt Focus-Ring nur bei Tastatur-Navigation

WCAG 2.4.3 (Focus Order): Links erscheinen in logischer Reihenfolge (left-to-right).

### aria-current für aktuelle Seite

Screenreader kündigen an: "Produkte, aktuelle Seite, Link" statt nur "Produkte, Link".

WCAG 1.3.1 (Info and Relationships): Visuelle Information (Farbe, Unterstreichung) wird programmatisch übermittelt.

### Kontrast-Anforderungen

Element	Farbe	Hintergrund	Kontrast	WCAG Level
Link Default	neutral.700	white	10.4:1	AAA
Link Hover	neutral.900	white	16.1:1	AAA
Link Active	blau.600	white	5.8:1	AA
Focus Outline	blau.300	white	3.2:1	AA (UI)

**WCAG 1.4.3 (Contrast Minimum):** Alle Text-Kontraste erfüllen mindestens AA (4.5:1 für Text).

**WCAG 1.4.11 (Non-text Contrast):** Focus Outline erfüllt 3:1 für UI-Komponenten.

## Skip-Link Pattern (optional)

Ermöglicht Tastatur-Nutzer\*innen, direkt zum Hauptinhalt zu springen.

```
<a href="#main-content" class="skip-link">
  Zum Hauptinhalt springen
</a>

<header class="header">
  <!-- Header Content -->
</header>

<main id="main-content">
  <!-- Page Content -->
</main>
```

```
.skip-link {
  position: absolute;
  top: -100px;
  left: 0;
  background: var(--hydrophon-blau-500);
  color: white;
  padding: 12px 16px;
  text-decoration: none;
  z-index: 9999;
}

.skip-link:focus {
  top: 0;
}
```

**Empfehlung:** Implementiere Skip-Links für Websites mit umfangreichen Headern oder komplexen Navigationen.

## Best Practices

### Inhaltliche Richtlinien

1. **Maximal 7 Hauptnavigationspunkte** Millers Gesetz: Menschen können  $7 \pm 2$  Informationseinheiten im Arbeitsgedächtnis behalten. Mehr Links führen zu Entscheidungslähmung.
2. **Klare, actionsorientierte Labels** ✓ "Produkte", "Support", "Kontakt" ✗ "Mehr erfahren", "Hier klicken", "Seite 1"
3. **Konsistente Benennung über alle Seiten** Wenn "Produkte" im Header steht, muss der `<title>` der Seite "Produkte" heißen, nicht "Produktübersicht".
4. **Logische Informationsarchitektur** Sortiere Links nach Nutzerpriorität: Home → Produkte → Support → Unternehmen → Kontakt

## Technische Richtlinien

1. Server-seitig `aria-current` setzen Verhindert Flash of Incorrect State bei JavaScript-Initialisierung.
2. Verwende semantisches HTML `<nav>` statt `<div role="navigation">` — semantische Elemente sind besser unterstützt.
3. Optimiere Logo-Asset SVG mit SVGO komprimieren, `alt="Hydrophon"` Text für Image Replacement.
4. Implementiere Preconnect für externe Fonts

```
<link rel="preconnect" href="https://fonts.googleapis.com">
```

5. Lazy-load Scroll-Shadow JavaScript Verwende Intersection Observer API statt Scroll-Events für bessere Performance.

## UX-Richtlinien

1. Sichtbares Feedback für alle Interaktionen Hover, Focus und Active-Zustände müssen visuell unterscheidbar sein.
  2. Keine plötzlichen Layout-Shifts Active-Indikator mit `position: absolute` verhindert Layout-Verschiebung.
  3. Touch-Targets mindestens 44x44px Link-Padding (12px + 16px × 2 = 40px) + Font-Size (16px) ≈ 44px vertikal.
- 

## Nicht verwenden

### ✗ Dropdown-Menüs auf Desktop

**Problem:** Verstecken Inhalt, benötigen komplexe JavaScript-Logik, sind schwierig barrierefrei umzusetzen.

**Alternative:** Flache Navigation mit max. 7 Links, detaillierte Unterseiten mit Breadcrumb-Navigation.

### ✗ Versteckte Navigation auf Desktop

**Problem:** "Hamburger-only" Header auf Desktop verwirren Nutzer\*innen, die horizontale Navigation erwarten.

**Alternative:** Vollständige Navigation auf Desktop, Hamburger nur auf Mobile.

### ✗ Auto-Hiding Header (versteckt beim Scrollen)

**Problem:** Nutzer\*innen verlieren Orientierung, müssen nach oben scrollen, um Navigation zu sehen.

**Alternative:** Sticky Header bleibt immer sichtbar.

### ✗ Animations-Exzesse

**Problem:** Slide-In-Animationen, Fad-in-Effekte und Parallax-Scrolling wirken unprofessionell im B2B-Kontext.

**Alternative:** Subtile 150ms Farbübergänge (`transition: color`), keine bewegten Elemente.

### ✗ JavaScript-abhängige Navigation

**Problem:** Wenn JavaScript nicht lädt, ist die Navigation unbrauchbar.

**Alternative:** HTML-Links funktionieren immer, JavaScript nur für Progressive Enhancement (Scroll-Shadow).

## Verwandte Komponenten

- **Mobile Drawer** — Mobile Navigation mit Hamburger-Button
  - **Breadcrumb** — Sekundäre Navigation für Hierarchie
  - **Footer** — Footer-Navigation mit Sitemap
- 

## Changelog

Version	Datum	Änderungen
1.0	2026-01-29	Initiale Dokumentation (Header Desktop Navigation)

---

## Mobile Navigation (Drawer)

Die Mobile Navigation verwendet ein Hamburger-Menü mit Slide-Out-Drawer Pattern. Dieses bewährte Pattern ermöglicht vollständige Navigation auf kleinen Bildschirmen ohne Viewport-Platz zu verschwenden.

**Anforderungen:** NAV-05 (Hamburger-Button), NAV-06 (Slide-Out-Drawer), NAV-07 (Focus-Management), NAV-08 (Keyboard-Accessibility)

## Übersicht

Das Mobile-Drawer-Pattern besteht aus zwei Komponenten:

1. Hamburger-Button — Toggle-Button im Header (nur sichtbar < 768px)
  2. Slide-Out-Drawer — Overlay-Panel mit vollständiger Navigation

#### Warum dieses Pattern:

- **Standardisiert** — Nutzer\*innen kennen Hamburger-Icon als Navigation-Indikator
  - **Voll zugänglich** — Mit korrektem ARIA und Focus-Management WCAG-konform
  - **Vollständige Navigation** — Alle Links sichtbar, kein versteckter Content
  - **Touch-optimiert** — Große Touch-Targets (48px Höhe) für Mobilgeräte

## B2B-Kontext:

Mobile B2B-Nutzer\*innen sind oft unterwegs (Baustelle, Lager, Außendienst). Die Navigation muss mit einer Hand bedienbar sein, schnell öffnen und klare Hierarchie zeigen.

Anatomie

Der Mobile-Drawer besteht aus fünf Elementen:



## 1. Hamburger-Button (Toggle)

**Funktion:** Öffnet/schließt den Drawer

**Element:** `<button>` mit `aria-expanded` und `aria-controls`

**Sizing:** 44×44px (WCAG AAA Touch-Target)

**Icon:** Lucide Menu (24px, 2px stroke)

## 2. Drawer-Container

**Funktion:** Enthält die vollständige Navigation

**Element:** `<div>` mit `role="dialog"`, `aria-modal="true"`

**Sizing:** 280px Breite, max 80vw (auf sehr kleinen Geräten)

**Position:** Fixed, links außerhalb des Viewports, slide-in via CSS Transform

## 3. Close-Button

**Funktion:** Schließt den Drawer

**Element:** `<button>` mit `aria-label`

**Icon:** Lucide X (24px, 2px stroke)

**Verhalten:** Erhält Fokus beim Öffnen des Drawers

## 4. Navigation-Links

**Funktion:** Gleiche Links wie Desktop-Header

**Element:** `<nav>` mit vertikaler Link-Liste

**Sizing:** 48px Mindesthöhe für Touch-Targets

**States:** Default, Hover, Active (`aria-current`), Focus

## 5. Backdrop-Overlay

**Funktion:** Visueller Kontext, schließt Drawer bei Klick

**Element:** `<div>` mit `rgba(0, 0, 0, 0.5)` Background

**Verhalten:** Fade-in/out mit Drawer, klickbar zum Schließen

---

## Hamburger-Button

### Visuelles Design

**Tokens:**

- Größe: `navigation.toggle.size` (44px)
- Icon-Größe: `navigation.toggle.iconSize` (24px)
- Farbe Default: `navigation.toggle.color.default` (neutral.700)
- Farbe Hover: `navigation.toggle.color.hover` (neutral.900)
- Hintergrund Default: `navigation.toggle.background.default` (transparent)
- Hintergrund Hover: `navigation.toggle.background.hover` (neutral.100)
- Border-Radius: `navigation.toggle.borderRadius` (4px)

## HTML-Struktur

```
<button
  class="header__toggle"
  aria-label="Menü öffnen"
  aria-expanded="false"
  aria-controls="mobile-drawer"
  id="drawer-toggle"
>
<svg
  width="24"
  height="24"
  viewBox="0 0 24 24"
  fill="none"
  stroke="currentColor"
  stroke-width="2"
  stroke-linecap="round"
  stroke-linejoin="round"
  aria-hidden="true"
>
  <line x1="3" y1="12" x2="21" y2="12" />
  <line x1="3" y1="6" x2="21" y2="6" />
  <line x1="3" y1="18" x2="21" y2="18" />
</svg>
</button>
```

## CSS

```
.header__toggle {
  display: none; /* Nur auf Mobile sichtbar */
  align-items: center;
  justify-content: center;
  width: var(--navigation-toggle-size);
  height: var(--navigation-toggle-size);
  padding: 0;
  border: none;
  background-color: var(--navigation-toggle-background-default);
  color: var(--navigation-toggle-color-default);
  border-radius: var(--navigation-toggle-border-radius);
  cursor: pointer;
  transition: background-color 150ms ease-in-out, color 150ms ease-in-out;
}

.header__toggle:hover {
  background-color: var(--navigation-toggle-background-hover);
  color: var(--navigation-toggle-color-hover);
}

.header__toggle:focus-visible {
  outline: var(--navigation-focus-outline-width) solid var(--navigation-focus-outline-color);
  outline-offset: var(--navigation-focus-outline-offset);
}

@media (max-width: 767px) {
  .header__toggle {
    display: flex;
  }
}
```

## ARIA-Attribute

Attribut	Wert (geschlossen)	Wert (offen)	Zweck
aria-label	"Menü öffnen"	"Menü schließen"	Benennt Button für Screenreader
aria-expanded	"false"	"true"	Kommuniziert Drawer-Status
aria-controls	"mobile-drawer"	"mobile-drawer"	Verknüpft Button mit gesteuertem Element

JavaScript-Update beim Toggle:

```
function toggleDrawer() {
  const isOpen = toggle.getAttribute('aria-expanded') === 'true';

  if (isOpen) {
    closeDrawer();
  } else {
    openDrawer();
  }
}
```

## Sichtbarkeit (Responsive)

Desktop (>= 768px): `display: none` — Desktop-Navigation ist im Header sichtbar

Mobile (< 768px): `display: flex` — Hamburger ersetzt horizontale Navigation

---

## Drawer-Verhalten

### Öffnen (Opening)

**Trigger:** Klick auf Hamburger-Button

**Schritte:**

1. Entferne `hidden`-Attribut vom Drawer
2. Setze `aria-hidden="false"` auf Drawer
3. Setze `aria-expanded="true"` auf Toggle-Button
4. Update Toggle `aria-label` zu "Menü schließen"
5. Bewege Fokus auf Close-Button
6. Aktiviere Fokus-Trap (Tab-Loop innerhalb Drawer)
7. Add Event-Listener für ESC-Taste

**CSS-Animation:** Transform von `translateX(-100%)` zu `translateX(0)`

**JavaScript:**

```
function openDrawer() {
  const drawer = document.getElementById('mobile-drawer');
  const toggle = document.getElementById('drawer-toggle');
  const closeButton = drawer.querySelector('.drawer__close');

  // Update Attribute
  drawer.removeAttribute('hidden');
  drawer.setAttribute('aria-hidden', 'false');
  toggle.setAttribute('aria-expanded', 'true');
  toggle.setAttribute('aria-label', 'Menü schließen');

  // Body Scroll Lock (verhindert Hintergrund-Scrollen)
  document.body.style.overflow = 'hidden';

  // Fokus auf Close-Button
  closeButton.focus();

  // Fokus-Trap aktivieren
  trapFocus(drawer);

  // ESC-Handler
  document.addEventListener('keydown', handleEscapeKey);
}
```

### Schließen (Closing)

**Trigger:**

- Klick auf Close-Button

- Klick auf Backdrop
- ESC-Taste
- Klick auf Navigation-Link (Navigation zu neuer Seite)

**Schritte:**

1. Setze `hidden`-Attribut auf Drawer
2. Setze `aria-hidden="true"` auf Drawer
3. Setze `aria-expanded="false"` auf Toggle-Button
4. Update Toggle `aria-label` zu "Menü öffnen"
5. Bewege Fokus zurück zum Toggle-Button
6. Deaktiviere Fokus-Trap
7. Remove Event-Listener für ESC-Taste

**CSS-Animation:** Transform von `translateX(0)` zu `translateX(-100%)`

**JavaScript:**

```
function closeDrawer() {
  const drawer = document.getElementById('mobile-drawer');
  const toggle = document.getElementById('drawer-toggle');

  // Update Attribute
  drawer.setAttribute('hidden', '');
  drawer.setAttribute('aria-hidden', 'true');
  toggle.setAttribute('aria-expanded', 'false');
  toggle.setAttribute('aria-label', 'Menü öffnen');

  // Body Scroll Unlock
  document.body.style.overflow = '';

  // Fokus zurück zum Toggle
  toggle.focus();

  // ESC-Handler entfernen
  document.removeEventListener('keydown', handleEscapeKey);
}

function handleEscapeKey(event) {
  if (event.key === 'Escape') {
    closeDrawer();
  }
}
```

**Animation**

**CSS Transform + Transition:**

```

.drawer {
  position: fixed;
  top: 0;
  left: 0;
  height: 100%;
  width: var(--navigation-drawer-width);
  max-width: var(--navigation-drawer-max-width);
  background-color: var(--navigation-drawer-background);
  box-shadow: var(--navigation-drawer-shadow);
  z-index: var(--navigation-drawer-z-index);
  transform: translateX(-100%);
  transition: transform var(--navigation-drawer-transition-duration) var(--navigation-drawer-transi
}

.drawer:not([hidden]) {
  transform: translateX(0);
}

.drawer__backdrop {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: var(--navigation-drawer-backdrop-color);
  z-index: calc(var(--navigation-drawer-z-index) - 1);
  opacity: 0;
  transition: opacity var(--navigation-drawer-transition-duration) var(--navigation-drawer-transiti
}

.drawer__backdrop:not([hidden]) {
  opacity: 1;
}

```

**Timing:** 300ms ease-in-out (aus Token `navigation.drawer.transition.duration`)

---

## Focus-Trap Implementation

Ein Focus-Trap hält den Keyboard-Fokus innerhalb des Drawers. Ohne Focus-Trap könnte Tab den Fokus auf Elemente hinter dem Drawer bewegen.

### Anforderungen

1. **Tab am Ende** wraps zum ersten fokussierbaren Element
2. **Shift+Tab am Anfang** wraps zum letzten fokussierbaren Element
3. Nur sichtbare, fokussierbare Elemente werden berücksichtigt

## Vollständige JavaScript-Implementation

```
function trapFocus(element) {
  const focusableElements = element.querySelectorAll(
    'button:not([disabled]), a[href], input:not([disabled]), select:not([disabled]), textarea:not([disabled])'
  );

  const firstFocusable = focusableElements[0];
  const lastFocusable = focusableElements[focusableElements.length - 1];

  function handleTabKey(event) {
    if (event.key !== 'Tab') return;

    if (event.shiftKey) {
      // Shift + Tab
      if (document.activeElement === firstFocusable) {
        event.preventDefault();
        lastFocusable.focus();
      }
    } else {
      // Tab
      if (document.activeElement === lastFocusable) {
        event.preventDefault();
        firstFocusable.focus();
      }
    }
  }

  element.addEventListener('keydown', handleTabKey);

  // Cleanup-Funktion für Event-Listener-Entfernung beim Schließen
  return () => {
    element.removeEventListener('keydown', handleTabKey);
  };
}
```

## Verwendung

```
let removeFocusTrap = null;

function openDrawer() {
  // ... andere Opening-Logik

  removeFocusTrap = trapFocus(drawer);
}

function closeDrawer() {
  // ... andere Closing-Logik

  if (removeFocusTrap) {
    removeFocusTrap();
    removeFocusTrap = null;
  }
}
```

## Navigation-Links im Drawer

### Layout

Navigation-Links werden vertikal gestapelt statt horizontal angeordnet.

#### Tokens:

- Link Padding X: `navigation.drawer.link.padding.x` (16px)
- Link Padding Y: `navigation.drawer.link.padding.y` (12px)
- Link Min-Height: `navigation.drawer.link.minLength` (48px)
- Link Gap: `navigation.drawer.link.gap` (8px)

### HTML-Struktur

```
<nav class="drawer__nav" aria-label="Hauptnavigation">
  <a href="/" class="drawer__link">Home</a>
  <a href="/produkte" class="drawer__link" aria-current="page">Produkte</a>
  <a href="/loesungen" class="drawer__link">lösungen</a>
  <a href="/support" class="drawer__link">Support</a>
  <a href="/unternehmen" class="drawer__link">Unternehmen</a>
  <a href="/kontakt" class="drawer__link">Kontakt</a>
</nav>
```

## CSS

```
.drawer__nav {
  display: flex;
  flex-direction: column;
  gap: var(--navigation-drawer-link-gap);
  padding: var(--navigation-drawer-padding);
}

.drawer__link {
  display: flex;
  align-items: center;
  min-height: var(--navigation-drawer-link-min-height);
  padding: var(--navigation-drawer-link-padding-y) var(--navigation-drawer-link-padding-x);
  color: var(--navigation-link-color-default);
  font-size: var(--navigation-link-font-size);
  font-weight: var(--navigation-link-font-weight-default);
  text-decoration: none;
  border-radius: var(--border-radius-base);
  transition: var(--navigation-link-transition);
}

.drawer__link:hover {
  color: var(--navigation-link-color-hover);
  background-color: var(--neutral-100);
}

.drawer__link[aria-current="page"] {
  color: var(--navigation-link-color-active);
  font-weight: var(--navigation-link-font-weight-active);
  background-color: var(--hydrophon-blau-50);
  border-left: 3px solid var(--navigation-active-indicator-color);
}

.drawer__link:focus-visible {
  outline: var(--navigation-focus-outline-width) solid var(--navigation-focus-outline-color);
  outline-offset: var(--navigation-focus-outline-offset);
}
```

## Zustände

Drawer-Links verwenden die gleichen Zustände wie Desktop-Navigation:

1. **Default** — neutral.700, medium weight
2. **Hover** — neutral.900 + neutral.100 Background
3. **Active (aria-current)** — blau.600, semibold, blau.50 Background, linker Border
4. **Focus** — 2px Outline blau.300

### Unterschied zum Desktop:

- Vertikales Layout statt horizontal
- Größere Touch-Targets (48px min-height)
- Hintergrundfarbe bei Hover/Active statt nur Textfarbe

## Tokens

Token-Name	Wert	Beschreibung
navigation.drawer.width	280px	Drawer feste Breite
navigation.drawer.maxWidth	80vw	Drawer maximale Breite (kleine Geräte)
navigation.drawer.background	#ffffff	Drawer Hintergrundfarbe (neutral.white)
navigation.drawer.shadow	0 20px 25px ...	Drawer Schatten (shadow.xl)
navigation.drawer.backdropColor	rgba(0,0,0,0.5)	Backdrop Overlay Farbe
navigation.drawer.zIndex	1000	Drawer z-Index (über Content)
navigation.drawer.transition.duration	300ms	Slide-Animation Dauer
navigation.drawer.transition.easing	ease-in-out	Slide-Animation Easing
navigation.drawer.padding	24px	Drawer inneres Padding (spacing.6)
navigation.drawer.link.padding.x	16px	Drawer Link horizontales Padding (spacing.4)
navigation.drawer.link.padding.y	12px	Drawer Link vertikales Padding (spacing.3)
navigation.drawer.link.minLength	48px	Drawer Link minimale Höhe (Touch-Target)
navigation.drawer.link.gap	8px	Vertikaler Abstand zwischen Links (spacing.2)
navigation.toggle.size	44px	Hamburger-Button Größe (WCAG AAA)
navigation.toggle.iconSize	24px	Hamburger-Icon Größe
navigation.toggle.color.default	#404040	Hamburger Standard-Farbe (neutral.700)
navigation.toggle.color.hover	#171717	Hamburger Hover-Farbe (neutral.900)
navigation.toggle.background.default	transparent	Hamburger Standard-Hintergrund
navigation.toggle.background.hover	#f5f5f5	Hamburger Hover-Hintergrund (neutral.100)
navigation.toggle.borderRadius	4px	Hamburger Eckradius (borderRadius.base)

CSS-Variable-Mapping:

```

navigation.drawer.width          → --navigation-drawer-width
navigation.drawer.transition.duration → --navigation-drawer-transition-duration
navigation.toggle.size          → --navigation-toggle-size

```

## Code-Beispiele

### Vollständiges HTML

```
<!-- Hamburger-Button (im Header) -->
<button
  class="header__toggle"
  aria-label="Menü öffnen"
  aria-expanded="false"
  aria-controls="mobile-drawer"
  id="drawer-toggle"
>
  <svg width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
    <line x1="3" y1="12" x2="21" y2="12" />
    <line x1="3" y1="6" x2="21" y2="6" />
    <line x1="3" y1="18" x2="21" y2="18" />
  </svg>
</button>

<!-- Drawer (außerhalb Header) -->
<div
  id="mobile-drawer"
  class="drawer"
  role="dialog"
  aria-modal="true"
  aria-label="Navigation"
  hidden
  aria-hidden="true"
>
  <!-- Close-Button -->
  <button class="drawer__close" aria-label="Menü schließen">
    <svg width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
      <line x1="18" y1="6" x2="6" y2="18" />
      <line x1="6" y1="6" x2="18" y2="18" />
    </svg>
  </button>

  <!-- Navigation-Links -->
  <nav class="drawer__nav" aria-label="Hauptnavigation">
    <a href="/" class="drawer__link">Home</a>
    <a href="/produkte" class="drawer__link" aria-current="page">Produkte</a>
    <a href="/loesungen" class="drawer__link">Lösungen</a>
    <a href="/support" class="drawer__link">Support</a>
    <a href="/unternehmen" class="drawer__link">Unternehmen</a>
    <a href="/kontakt" class="drawer__link">Kontakt</a>
  </nav>
</div>

<!-- Backdrop -->
<div
  id="drawer-backdrop"
  class="drawer__backdrop"
  hidden
>
```

```
    aria-hidden="true"
></div>
```

## Vollständiges CSS

```
/* Drawer Container */
.drawer {
  position: fixed;
  top: 0;
  left: 0;
  height: 100%;
  width: var(--navigation-drawer-width);
  max-width: var(--navigation-drawer-max-width);
  background-color: var(--navigation-drawer-background);
  box-shadow: var(--navigation-drawer-shadow);
  z-index: var(--navigation-drawer-z-index);
  overflow-y: auto;
  transform: translateX(-100%);
  transition: transform var(--navigation-drawer-transition-duration) var(--navigation-drawer-transi
}

.drawer:not([hidden]) {
  transform: translateX(0);
}

/* Close-Button */
.drawer__close {
  position: absolute;
  top: var(--spacing-4);
  right: var(--spacing-4);
  display: flex;
  align-items: center;
  justify-content: center;
  width: 44px;
  height: 44px;
  padding: 0;
  border: none;
  background-color: transparent;
  color: var(--neutral-700);
  border-radius: var(--border-radius-base);
  cursor: pointer;
  transition: background-color 150ms ease-in-out, color 150ms ease-in-out;
}

.drawer__close:hover {
  background-color: var(--neutral-100);
  color: var(--neutral-900);
}

.drawer__close:focus-visible {
  outline: var(--navigation-focus-outline-width) solid var(--navigation-focus-outline-color);
  outline-offset: var(--navigation-focus-outline-offset);
}

/* Navigation */
.drawer__nav {
  display: flex;
  flex-direction: column;
  gap: var(--navigation-drawer-link-gap);
  padding: var(--navigation-drawer-padding);
```

```
margin-top: 60px; /* Raum für Close-Button */  
}  
  
.drawer__link {  
  display: flex;  
  align-items: center;  
  min-height: var(--navigation-drawer-link-min-height);  
  padding: var(--navigation-drawer-link-padding-y) var(--navigation-drawer-link-padding-x);  
  color: var(--navigation-link-color-default);  
  font-size: var(--navigation-link-font-size);  
  font-weight: var(--navigation-link-font-weight-default);  
  text-decoration: none;  
  border-radius: var(--border-radius-base);  
  transition: var(--navigation-link-transition), background-color 150ms ease-in-out;  
}  
  
.drawer__link:hover {  
  color: var(--navigation-link-color-hover);  
  background-color: var(--neutral-100);  
}  
  
.drawer__link[aria-current="page"] {  
  color: var(--navigation-link-color-active);  
  font-weight: var(--navigation-link-font-weight-active);  
  background-color: var(--hydrophon-blau-50);  
  border-left: 3px solid var(--navigation-active-indicator-color);  
}  
  
.drawer__link:focus-visible {  
  outline: var(--navigation-focus-outline-width) solid var(--navigation-focus-outline-color);  
  outline-offset: var(--navigation-focus-outline-offset);  
}  
  
/* Backdrop */  
.drawer__backdrop {  
  position: fixed;  
  top: 0;  
  left: 0;  
  width: 100%;  
  height: 100%;  
  background-color: var(--navigation-drawer-backdrop-color);  
  z-index: calc(var(--navigation-drawer-z-index) - 1);  
  opacity: 0;  
  transition: opacity var(--navigation-drawer-transition-duration) var(--navigation-drawer-transition-ease);  
  cursor: pointer;  
}  
  
.drawer__backdrop:not([hidden]) {  
  opacity: 1;  
}
```

## Vollständiges JavaScript

```
// Elemente
const toggle = document.getElementById('drawer-toggle');
const drawer = document.getElementById('mobile-drawer');
const backdrop = document.getElementById('drawer-backdrop');
const closeButton = drawer.querySelector('.drawer__close');

let removeFocusTrap = null;

// Event-Listener
toggle.addEventListener('click', toggleDrawer);
closeButton.addEventListener('click', closeDrawer);
backdrop.addEventListener('click', closeDrawer);

// Toggle-Funktion
function toggleDrawer() {
  const isOpen = toggle.getAttribute('aria-expanded') === 'true';
  isOpen ? closeDrawer() : openDrawer();
}

// Drawer öffnen
function openDrawer() {
  drawer.removeAttribute('hidden');
  drawer.setAttribute('aria-hidden', 'false');
  backdrop.removeAttribute('hidden');
  backdrop.setAttribute('aria-hidden', 'false');

  toggle.setAttribute('aria-expanded', 'true');
  toggle.setAttribute('aria-label', 'Menü schließen');

  document.body.style.overflow = 'hidden';

  closeButton.focus();

  removeFocusTrap = trapFocus(drawer);

  document.addEventListener('keydown', handleEscapeKey);
}

// Drawer schließen
function closeDrawer() {
  drawer.setAttribute('hidden', '');
  drawer.setAttribute('aria-hidden', 'true');
  backdrop.setAttribute('hidden', '');
  backdrop.setAttribute('aria-hidden', 'true');

  toggle.setAttribute('aria-expanded', 'false');
  toggle.setAttribute('aria-label', 'Menü öffnen');

  document.body.style.overflow = '';

  toggle.focus();

  if (removeFocusTrap) {
    removeFocusTrap();
    removeFocusTrap = null;
  }
}
```

```

    }

    document.removeEventListener('keydown', handleEscapeKey);
}

// ESC-Taste Handler
function handleEscapeKey(event) {
  if (event.key === 'Escape') {
    closeDrawer();
  }
}

// Focus-Trap
function trapFocus(element) {
  const focusableElements = element.querySelectorAll(
    'button:not([disabled]), a[href], input:not([disabled]), select:not([disabled]), textarea:not([disabled])'
  );

  const firstFocusable = focusableElements[0];
  const lastFocusable = focusableElements[focusableElements.length - 1];

  function handleTabKey(event) {
    if (event.key !== 'Tab') return;

    if (event.shiftKey) {
      if (document.activeElement === firstFocusable) {
        event.preventDefault();
        lastFocusable.focus();
      }
    } else {
      if (document.activeElement === lastFocusable) {
        event.preventDefault();
        firstFocusable.focus();
      }
    }
  }

  element.addEventListener('keydown', handleTabKey);

  return () => {
    element.removeEventListener('keydown', handleTabKey);
  };
}

```

## Barrierefreiheit

### WCAG 2.4.3: Focus Order

Fokus-Reihenfolge im Drawer:

1. Close-Button (erhält Fokus beim Öffnen)
2. Navigation-Link 1
3. Navigation-Link 2
4. ...

5. Navigation-Link N
6. (Tab wraps zu Close-Button)

## WCAG 2.1.1: Keyboard Accessible

Anforderungen erfüllt:

- Drawer öffnet per Enter/Space auf Toggle-Button
- Drawer schließt per ESC-Taste
- Drawer schließt per Enter/Space auf Close-Button
- Tab-Navigation innerhalb Drawer
- Backdrop schließt per Click (optional, nicht Keyboard)

## WCAG 2.1.2: No Keyboard Trap

Focus-Trap hält Fokus im Drawer, ABER:

- ESC-Taste schließt Drawer und gibt Fokus frei
- Close-Button ist immer fokussierbar
- Fokus kehrt zu Toggle-Button zurück (kein Verlust)

## ARIA-Attribute

Element	ARIA-Attribut	Wert/Zweck
Drawer	<code>role="dialog"</code>	Identifiziert als Dialog-Komponente
Drawer	<code>aria-modal="true"</code>	Kommuniziert modales Verhalten
Drawer	<code>aria-label</code>	"Navigation" — Benennt Dialog
Drawer	<code>aria-hidden</code>	"true" (geschlossen) / "false" (offen)
Toggle	<code>aria-expanded</code>	"false" (geschlossen) / "true" (offen)
Toggle	<code>aria-controls</code>	ID des Drawers — Verknüpfung
Toggle	<code>aria-label</code>	"Menü öffnen" / "Menü schließen" — Beschreibung
Close-Button	<code>aria-label</code>	"Menü schließen" — Beschreibung

## Screen Reader Ankündigungen

**Beim Öffnen:**

"Dialog, Navigation. Menü schließen, Button."

**Beim Schließen:**

"Menü öffnen, Button."

**Bei Navigation-Link:**

"Produkte, aktuelle Seite, Link."

---

## Best Practices

### UX-Richtlinien

1. **Close-Button erhält Fokus beim Öffnen** Screenreader-Nutzer\*innen hören sofort, dass ein Dialog geöffnet wurde und wie man ihn schließt.
2. **Immer ESC zum Schließen** Keyboard-Nutzer\*innen erwarten ESC-Funktionalität bei modalen Overlays.
3. **Backdrop klickbar** Maus-Nutzer\*innen erwarten, dass Klick außerhalb des Drawers diesen schließt.
4. **Body-Scroll-Lock aktivieren** Verhindert verwirrende Situation, wenn Hintergrund scrollt, während Drawer offen ist.
5. **Fokus-Rückkehr zum Toggle** Nutzer\*innen wissen nach dem Schließen, wo sie waren.

### Technische Richtlinien

1. **hidden statt display: none (JavaScript)** `hidden` -Attribut ist semantisch korrekter und kombiniert gut mit CSS `[hidden]`.
2. **aria-hidden synchron mit hidden** Beide Attribute müssen immer zusammen gesetzt werden.
3. **Transform statt left/right für Animation** Transform ist GPU-beschleunigt, performanter als Layout-Shifts.
4. **z-index Hierarchie** Backdrop: 999, Drawer: 1000 (damit Drawer über Backdrop liegt).
5. **Event-Listener Cleanup** Entferne ESC-Listener beim Schließen, um Memory Leaks zu vermeiden.

### Accessibility-Richtlinien

1. **role="dialog" + aria-modal="true"** Kommuniziert, dass Drawer modalen Kontext erzeugt (Hintergrund nicht interagierbar).
  2. **aria-label auf Drawer** Benennt Dialog für Screenreader (nicht nur visuell erkennbar).
  3. **Focus-Trap ist Pflicht** Ohne Focus-Trap können Tastatur-Nutzer\*innen hinter den Drawer gelangen.
  4. **Close-Button immer sichtbar** Auch wenn visuell klar ist, wie man schließt — Screenreader brauchen Button.
- 

## Nicht verwenden

### ✗ display: none ohne aria-hidden

**Problem:** Element ist visuell versteckt, aber für Screenreader noch erreichbar.

**Alternative:** Immer beide Attribute setzen: `hidden + aria-hidden="true"`.

### ✗ Fokus-Verlust beim Schließen

**Problem:** Fokus bleibt im geschlossenen (versteckten) Drawer, Tastatur-Nutzer\*innen wissen nicht, wo sie sind.

**Alternative:** Fokus immer zurück zum Toggle-Button bewegen.

### ✗ Fehlende ESC-Unterstützung

**Problem:** WCAG 2.1.2 verlangt Keyboard-Escape aus modalen Dialogen.

**Alternative:** Immer ESC-Listener implementieren.

## ✖ Backdrop nicht klickbar

**Problem:** Maus-Nutzer\*innen erwarten, dass Klick außerhalb schließt.

**Alternative:** Click-Event-Listener auf Backdrop.

## ✖ Kein Body-Scroll-Lock

**Problem:** Hintergrund scrollt, während Drawer offen ist — verwirrend und visuell unschön.

**Alternative:** `document.body.style.overflow = 'hidden'` beim Öffnen.

## ✖ Animationen ohne prefers-reduced-motion

**Problem:** Nutzer\*innen mit vestibulären Störungen können durch Animationen desorientiert werden.

**Alternative:**

```
@media (prefers-reduced-motion: reduce) {
  .drawer,
  .drawer__backdrop {
    transition: none;
  }
}
```

## Verwandte Komponenten

- [Header Navigation](#) — Desktop Header mit horizontaler Navigation
- [Breadcrumb](#) — Sekundäre Navigation für Hierarchie

## Changelog

Version	Datum	Änderungen
1.0	2026-01-29	Initiale Dokumentation (Mobile Navigation Drawer)

# Breadcrumb-Navigation

---

Breadcrumbs zeigen Nutzern ihre aktuelle Position in der Seitenhierarchie und ermöglichen schnelle Navigation zu übergeordneten Ebenen.

## Übersicht

**Zweck:** Breadcrumbs bieten Orientierung auf Websites mit komplexer Informationsarchitektur und mehreren Hierarchieebenen.

**Wann verwenden:**

- Seiten mit 3 oder mehr Hierarchieebenen (z.B. Home → Produkte → Kategorie → Produkt)
- Externe Einstiege über Suchmaschinen oder direkte Links
- B2B-Kontext mit verschachtelten Produktkategorien oder Unternehmensstrukturen

**Wann nicht verwenden:**

- Flache Seitenstrukturen (nur 1-2 Ebenen)
- Single-Page-Applications ohne Hierarchie
- Formulare oder Step-by-Step-Prozesse (verwenden Sie stattdessen Stepper/Wizard)

## Anatomie

Eine Breadcrumb-Navigation besteht aus folgenden Elementen:

1. **Container:** `<nav>` Element mit `aria-label="Breadcrumbs"` für Screen Reader
2. **Liste:** `<ol>` (ordered list) für semantische Reihenfolge
3. **Listenelemente:** `<li>` für jede Ebene
4. **Links:** `<a>` für navigierbare Ebenen
5. **Aktuelle Seite:** `<span>` (kein Link) mit `aria-current="page"`
6. **Separator:** Visuelles Trennzeichen zwischen Ebenen (CSS `::after`)



## Separator

### Chevron (>) - Empfohlen

Der Chevron ist das empfohlene Separator-Zeichen für Breadcrumbs im Hydrophon Design System.

**Vorteile:**

- Modern und richtungsweisend

- Klare visuelle Hierarchie von links nach rechts
- Konsistent mit Button-Icons (Lucide ChevronRight)
- Internationale Verwendbarkeit (keine Sprachabhängigkeit)

Technische Umsetzung:

```
.breadcrumb li::after {
  content: ">";
  color: var(--breadcrumb-separator-color);
  margin: 0 var(--breadcrumb-separator-marginX);
}

.breadcrumb li:last-child::after {
  content: none;
}
```

**Wichtig:** Der Separator wird als CSS Pseudo-Element ( ::after ) eingefügt und ist damit automatisch für Screen Reader versteckt. Assistive Technologien erkennen die Navigation durch die semantische <ol>-Struktur.

## Alternative Separatoren

Slash (/):

- Klassisch und weit verbreitet
- Gut für technische oder datei-basierte Hierarchien
- Beispiel: Home / Produkte / Kategorie

Pfeil (→):

- Stark richtungsweisend
- Moderner als Slash, weniger subtil als Chevron
- Beispiel: Home → Produkte → Kategorie

**Token-basiert:** Alle Separatoren können über `breadcrumb.separator.content` zentral geändert werden.

## Mobile-Verhalten

Breadcrumbs benötigen auf kleinen Bildschirmen besondere Aufmerksamkeit, um Lesbarkeit und Bedienbarkeit zu gewährleisten.

### Option 1: Vollständige Breadcrumbs mit Wrap

Die einfachste Lösung: Breadcrumbs umbrechen bei Platzmangel.

**Vorteile:**

- Vollständige Hierarchie sichtbar
- Keine JavaScript-Logik erforderlich
- Barrierefreundlich

**Nachteile:**

- Bei vielen Ebenen vertikal sehr lang
- Kann mobilen Viewport dominieren

```
@media (max-width: 640px) {
  .breadcrumb ol {
    flex-wrap: wrap;
  }
}
```

### Option 2: Gekürzte Breadcrumbs (Home + Aktuell)

Bei mehr als 3 Ebenen: Nur erste und letzte Ebene anzeigen.

#### Vorteile:

- Kompakt und übersichtlich
- Wichtigste Information (Home + Aktuell) sichtbar

#### Nachteile:

- Mittlere Ebenen nicht direkt erreichbar
- Benötigt JavaScript oder CSS-Logik

```
@media (max-width: 640px) {
  /* Verstecke mittlere Elemente */
  .breadcrumb li:not(:first-child):not(:last-child) {
    display: none;
  }

  /* Zeige Ellipsis-Indikator */
  .breadcrumb li:first-child::after {
    content: ">";
    margin: 0 var(--breadcrumb-separator-marginX);
  }
}
```

### Empfehlung

- < 4 Ebenen: Vollständige Breadcrumbs mit Wrap
- ≥ 4 Ebenen: Gekürzte Breadcrumbs (Home + Aktuell)
- Alternative: Horizontales Scrollen (touch-freundlich, aber weniger üblich)

### Tokens

Alle Breadcrumb-Styles sind über Design Tokens konfigurierbar:

Token	Wert	CSS-Variable	Beschreibung
breadcrumb.fontSize	{fontSize.sm} (14px)	--breadcrumb-fontSize	Kompakte Schriftgröße für sekundäre Navigation
breadcrumb.gap	{spacing.2} (8px)	--breadcrumb-gap	Abstand zwischen Breadcrumb-Elementen
breadcrumb.marginY	{spacing.4} (16px)	--breadcrumb-marginY	Vertikaler Abstand der Breadcrumb-Navigation
breadcrumb.separator.content	--breadcrumb-separator-content	Chevron-Separator	
breadcrumb.separator.color	{neutral.400}	--breadcrumb-separator-color	Dezente Separator-Farbe
breadcrumb.separator.marginX	{spacing.2} (8px)	--breadcrumb-separator-marginX	Horizontaler Abstand um Separator
breadcrumb.link.color	{hydrophon.blau.600}	--breadcrumb-link-color	Link-Farbe (etwas dunkler als primär)
breadcrumb.link.colorHover	{hydrophon.blau.700}	--breadcrumb-link-colorHover	Link-Farbe bei Hover
breadcrumb.link.textDecoration	none	--breadcrumb-link-textDecoration	Keine Unterstreichung im Standard
breadcrumb.link.textDecorationHover	underline	--breadcrumb-link-textDecorationHover	Unterstreichung bei Hover
breadcrumb.current.color	{neutral.900}	--breadcrumb-current-color	Aktuelle Seite in neutraler Farbe
breadcrumb.current.fontWeight	{fontWeight.medium} (500)	--breadcrumb-current-fontWeight	Medium-Gewicht hebt aktuelle Seite hervor

**Token-Referenzen:** Siehe `design-system/tokens/navigation.json`

## Code-Beispiele

### HTML

Vollständige semantische Struktur mit ARIA-Attributnen:

```
<nav aria-label="Breadcrumbs">
  <ol class="breadcrumb">
    <li>
      <a href="/">Home</a>
    </li>
    <li>
      <a href="/produkte">Produkte</a>
    </li>
    <li>
      <a href="/produkte/sanitaeranlagen">Sanitäranlagen</a>
    </li>
    <li>
      <span aria-current="page">Waschtische</span>
    </li>
  </ol>
</nav>
```

#### Wichtige Details:

- `<nav>` mit `aria-label="Breadcrumbs"` kennzeichnet Breadcrumb-Landmark
- `<ol>` statt `<ul>` für semantische Reihenfolge
- Links für alle navigierbaren Ebenen
- `<span>` (kein Link) für aktuelle Seite
- `aria-current="page"` auf aktueller Seite (zwingend erforderlich)

#### CSS

Vollständiges Styling mit Token-Referenzen:

```

/* Container */
.breadcrumb {
  margin: var(--breadcrumb-marginY) 0;
}

/* Liste */
.breadcrumb ol {
  display: flex;
  flex-wrap: wrap;
  gap: var(--breadcrumb-gap);
  list-style: none;
  padding: 0;
  margin: 0;
  font-size: var(--breadcrumb-fontSize);
}

/* Listenelemente */
.breadcrumb li {
  display: flex;
  align-items: center;
}

/* Separator (::after) */
.breadcrumb li::after {
  content: var(--breadcrumb-separator-content);
  color: var(--breadcrumb-separator-color);
  margin: 0 var(--breadcrumb-separator-marginX);
}

/* Kein Separator nach letztem Element */
.breadcrumb li:last-child::after {
  content: none;
}

/* Links */
.breadcrumb a {
  color: var(--breadcrumb-link-color);
  text-decoration: var(--breadcrumb-link-textDecoration);
  transition: color 150ms ease-in-out;
}

.breadcrumb a:hover {
  color: var(--breadcrumb-link-colorHover);
  text-decoration: var(--breadcrumb-link-textDecorationHover);
}

/* Focus-Indicator (konsistent mit globalen Link-Styles) */
.breadcrumb a:focus-visible {
  outline: 2px solid var(--hydrophon-blau-300);
  outline-offset: 2px;
  border-radius: 2px;
}

/* Aktuelle Seite */
.breadcrumb [aria-current="page"] {
  color: var(--breadcrumb-current-color);
  font-weight: var(--breadcrumb-current-fontWeight);
}

```

```

}

/* Mobile Anpassung: Gekürzte Breadcrumbs */
@media (max-width: 640px) {
  .breadcrumb li:not(:first-child):not(:last-child) {
    display: none;
  }
}

```

## React-Beispiel

```

function Breadcrumb({ items }) {
  return (
    <nav aria-label="Breadcrumbs">
      <ol className="breadcrumb">
        {items.map((item, index) => (
          <li key={item.href || index}>
            {index === items.length - 1 ? (
              <span aria-current="page">{item.label}</span>
            ) : (
              <a href={item.href}>{item.label}</a>
            )}
          </li>
        ))}
      </ol>
    </nav>
  );
}

// Verwendung
<Breadcrumb
  items={[
    { label: "Home", href: "/" },
    { label: "Produkte", href: "/produkte" },
    { label: "Sanitäranlagen", href: "/produkte/sanitaeranlagen" },
    { label: "Waschtische" } // kein href = aktuelle Seite
  ]}
/>

```

## Barrierefreiheit

Breadcrumbs erfüllen wichtige WCAG-Kriterien für Orientierung und Navigation.

### WCAG 2.4.8: Location (AAA)

**Kriterium:** Nutzer können ihre Position in der Website-Struktur erkennen.

**Erfüllung:** Breadcrumbs zeigen die vollständige Hierarchie und ermöglichen Navigation zu übergeordneten Ebenen.

**Hinweis:** WCAG 2.4.8 ist Level AAA (nicht AA). Breadcrumbs sind dennoch Best Practice für komplexe B2B-Websites.

### aria-current="page"

**Pflicht-Attribut:** Die aktuelle Seite muss mit `aria-current="page"` gekennzeichnet sein.

**Screen Reader-Ansage:** NVDA/JAWS kündigen an: "Waschtische, aktuelle Seite" statt nur "Waschtische".

**Anti-Pattern:** Aktuelle Seite als Link mit `class="active"` ist unzureichend - Screen Reader erkennen keinen Unterschied.

## Separator-Barrierefreiheit

**CSS Pseudo-Element:** Der Separator wird als `::after` Pseudo-Element eingefügt und ist automatisch für Screen Reader versteckt.

**Warum wichtig:** Screen Reader würden HTML-Separatoren vorlesen ("Home, Pfeil, Produkte, Pfeil, ..."). Das ist redundant und störend.

Richtig:

```
.breadcrumb li::after {  
    content: ">";  
}
```

Falsch:

```
<li><a href="/">Home</a> <span class="separator">></span></li>
```

## Kontrast

**WCAG 1.4.3:** Text benötigt 4.5:1 Kontrast.

Erfüllung:

- Links: `hydrophon.blau.600` (#007bb8) auf Weiß = 4.53:1 ✓
- Separator: `neutral.400` = 3:1 (akzeptabel für UI-Elemente, nicht Text)
- Aktuelle Seite: `neutral.900` = 16.5:1 ✓

## Keyboard-Navigation

**Tab:** Fokus springt von Link zu Link (Separator werden übersprungen).

**Enter:** Aktiviert fokussierten Link.

**Aktuelle Seite:** Nicht fokussierbar (kein Link), korrekt.

## Focus-Indicator

**Standard:** Konsistent mit globalen Link-Styles (2px outline, 2px offset).

**WCAG 2.4.7:** Focus muss visuell erkennbar sein - 3:1 Kontrast zum Hintergrund.

Erfüllung: `hydrophon.blau.300` (#5cc2f1) auf Weiß = 3.07:1 ✓

## Best Practices

### Semantisches HTML

Verwenden:

- `<nav>` mit `aria-label="Breadcrumbs"` für Landmark

- `<ol>` statt `<ul>` für Reihenfolge
- `aria-current="page"` auf aktueller Seite
- Separator als CSS `::after` Pseudo-Element

**Nicht verwenden:**

- `<div>` statt `<nav>`
- `<ul>` statt `<ol>`
- Link auf aktuelle Seite (auch nicht mit `disabled`)
- Separator als HTML-Element

## Konsistenz

**Position:** Breadcrumbs sollten immer an derselben Stelle erscheinen:

- Direkt unter Header/Navigation
- Über dem Hauptinhalt (`h1`)
- Links ausgerichtet (konsistent mit Inhalts-Container)

**Reihenfolge:** Immer von allgemein zu spezifisch (Home → Kategorie → Unterkategorie → Seite).

## Mobile First

**Design-Entscheidung früh treffen:**

- Vollständig mit Wrap oder gekürzt?
- Testen mit realen Hierarchien (nicht nur 3 Ebenen)
- Touch-Targets beachten (links sollten mindestens 44px hoch sein)

**Nicht:** Breadcrumbs auf Mobile komplett verstecken - sie sind gerade bei externen Einstiegen wertvoll.

## Integration mit Header

Breadcrumbs werden typischerweise zwischen Header-Navigation und Hauptinhalt positioniert.

## Layout-Beispiel

```

<header>
  <div class="header-container">
    <a href="/" class="logo">Hydrophon</a>
    <nav aria-label="Hauptnavigation">
      <!-- Primäre Navigation -->
    </nav>
  </div>
</header>

<!-- Breadcrumbs unterhalb Header -->
<nav aria-label="Breadcrumbs" class="breadcrumb-container">
  <ol class="breadcrumb">
    <!-- Breadcrumb-Items -->
  </ol>
</nav>

<main>
  <h1>Aktuelle Seite</h1>
  <!-- Hauptinhalt -->
</main>

```

## Container-Breite

**Empfehlung:** Breadcrumbs sollten die gleiche maximale Breite wie der Hauptinhalt verwenden.

```

.breadcrumb-container {
  max-width: var(--container-xl); /* 1280px */
  margin: 0 auto;
  padding: 0 var(--spacing-6);
}

```

## Visueller Bezug

### Option 1: Dezent

- Breadcrumbs ohne Hintergrundfarbe
- Dezentes Grau für Links
- Minimale visuelle Präsenz

### Option 2: Hervorgehoben (aktuell)

- Leichter Hintergrund ( neutral.50 )
- Border unten zur Trennung
- Breadcrumbs als eigener Bereich erkennbar

## Nicht verwenden

### Link auf aktuelle Seite

Falsch:

```
<li>
  <a href="/aktuelle-seite" class="active" aria-current="page">
    Aktuelle Seite
  </a>
</li>
```

**Problem:** Links auf die aktuelle Seite sind verwirrend und redundant.

Richtig:

```
<li>
  <span aria-current="page">Aktuelle Seite</span>
</li>
```

## Separator als HTML-Element

Falsch:

```
<li><a href="/">Home</a></li>
<li class="separator">></li>
<li><a href="/produkte">Produkte</a></li>
```

**Problem:**

- Screen Reader lesen Separator vor
- Listenstruktur wird gestört
- Unnötige DOM-Elemente

Richtig: CSS ::after Pseudo-Element (siehe Code-Beispiele).

## Fehlende aria-label auf nav

Falsch:

```
<nav>
  <ol class="breadcrumb">...</ol>
</nav>
```

**Problem:** Screen Reader kündigen an "Navigation" - aber welche? Header? Footer? Breadcrumbs?

Richtig:

```
<nav aria-label="Breadcrumbs">
  <ol class="breadcrumb">...</ol>
</nav>
```

## JavaScript-basierte Breadcrumbs ohne Fallback

**Falsch:** Breadcrumbs nur per JavaScript rendern ohne Server-Side-Rendering.

**Problem:** SEO-Nachteile, langsamer Time-to-Interactive, potenzielle Fehler bei JavaScript-Problemen.

**Richtig:** Breadcrumbs server-seitig oder mit Static Site Generation rendern.

## Referenzen

**Header-Navigation:** Siehe `design-system/docs/navigation/header.md` für konsistente Link-Styles und Focus-Indicator.

**Token-System:** Siehe `design-system/tokens/navigation.json` für alle Breadcrumb-Tokens.

**ARIA-Pattern:** [W3C WAI-ARIA Authoring Practices - Breadcrumb](#)

**WCAG 2.4.8:** [Understanding SC 2.4.8: Location \(Level AAA\)](#)

---

# Footer

Der Footer bietet sekundäre Navigation, Kontaktinformationen und rechtlich erforderliche Links am Ende jeder Seite.

## Übersicht

**Zweck:** Der Footer ist die letzte Orientierungshilfe auf einer Seite und vermittelt Vertrauen durch Vollstndigkeit und Professionalitt.

**B2B-Kontext:** Im B2B-Bereich ist der Footer besonders wichtig für:

- Kontaktinformationen (Vertrieb, Support, Service-Hotlines)
  - Rechtliche Pflichtangaben (Impressum, Datenschutz, AGB)
  - Sekundäre Navigation (Über uns, Karriere, Presse)
  - Vertrauensbildende Elemente (Zertifikate, Partner, Social Media)

Wann verwenden:

- Auf jeder öffentlichen Seite der Website
  - Konsistente Darstellung über alle Seiten hinweg

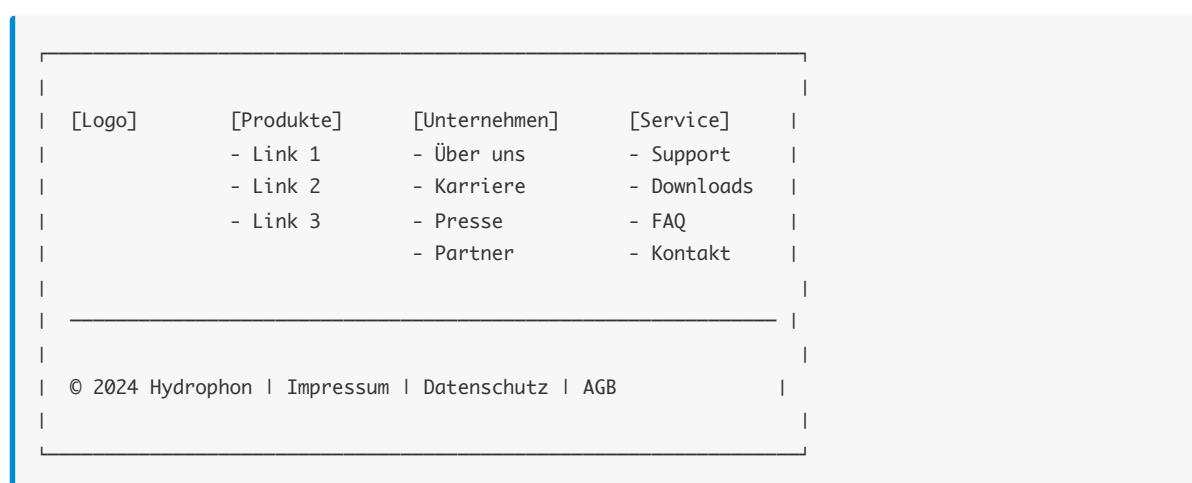
#### Wann nicht verwenden:

- Single-Page-Applications ohne Footer-Anforderung
  - Formulare oder Checkout-Prozesse (vereinfachter Footer möglich)

Anatomie

Ein Footer besteht typischerweise aus folgenden Bereichen:

- 1. Footer-Container:** <footer> Element mit automatischem `role="contentinfo"`
  - 2. Logo (optional):** Markenidentität und Link zur Startseite
  - 3. Link-Gruppen:** 3-4 thematische Spalten mit Überschriften und Links
  - 4. Kontakt (optional):** Adresse, Telefon, E-Mail in <address> Element
  - 5. Legal-Bereich:** Copyright, Impressum, Datenschutz, AGB



## Layout

### Desktop (>= 768px)

Auf Desktop-Geräten wird ein mehrspaliges Grid-Layout verwendet:

#### Grid-Struktur:

- 3-4 Spalten für Link-Gruppen
- Optional: Logo links (zusätzliche Spalte oder über Link-Gruppen)
- Legal-Bereich volle Breite unten

#### Vorteile:

- Übersichtliche Gruppierung verwandter Links
- Effiziente Platznutzung
- Schnelles Scannen durch klare Struktur

```
.footer-content {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: var(--footer-gap); /* 32px */
}
```

### Tablet (768px - 1024px)

**Option 1:** 3 Spalten (eine Gruppe bricht um) **Option 2:** 2 Spalten (je 2 Gruppen)

```
@media (min-width: 768px) and (max-width: 1023px) {
  .footer-content {
    grid-template-columns: repeat(3, 1fr);
  }
}
```

### Mobile (< 768px)

**Einspaltig:** Alle Link-Gruppen untereinander gestapelt.

#### Option 1: Gestackt (empfohlen)

- Alle Gruppen permanent sichtbar
- Einfache Implementierung
- Kein JavaScript erforderlich

#### Option 2: Akkordeon

- Überschriften klappbar
- Kompakter initial
- Benötigt JavaScript + ARIA-Attribute

**Empfehlung:** Gestackt für < 6 Link-Gruppen, Akkordeon für ≥ 6 Gruppen.

```
@media (max-width: 767px) {
  .footer-content {
    grid-template-columns: 1fr;
  }
}
```

## Link-Gruppen

### Typische Gruppierung

Für eine B2B-Website wie Hydrophon empfehlen sich folgende Link-Gruppen:

#### Gruppe 1: Produkte

- Produktkategorien (z.B. Waschtische, Duschen, Armaturen)
- Produktlinien (Hydrophon, Gluy)
- Zubehör

#### Gruppe 2: Unternehmen

- Über uns
- Karriere
- Nachhaltigkeit
- Presse
- Partner

#### Gruppe 3: Service

- Support
- Downloads (Kataloge, Technische Datenblätter)
- FAQ
- Garantie
- Händlersuche

#### Gruppe 4: Kontakt

- Kontaktformular
- Service-Hotline
- Vertrieb
- Showrooms
- Adresse (falls in <address> Element)

**Anzahl:** Maximum 4-5 Gruppen für gute Übersichtlichkeit.

## Styling

### Überschriften:

- Uppercase für visuelle Hierarchie
- Semibold (600) für Betonung
- 14px für kompakte Darstellung
- 16px Abstand nach unten

```
.footer-heading {
  font-size: var(--footer-heading-fontSize); /* 14px */
  font-weight: var(--footer-heading-fontWeight); /* 600 */
  color: var(--footer-heading-color); /* neutral.900 */
  text-transform: var(--footer-heading-textTransform); /* uppercase */
  letter-spacing: var(--footer-heading-letterSpacing); /* 0.05em */
  margin-bottom: var(--footer-heading-marginBottom); /* 16px */
}
```

**Links:**

- Dezentes Grau ( `neutral.600` ) für sekundäre Navigation
- Dunkler bei Hover ( `neutral.900` )
- 14px Schriftgröße
- line-height 2 für großzügige Touch-Targets (28px bei 14px Schrift)

```
.footer-link {
  color: var(--footer-link-color); /* neutral.600 */
  font-size: var(--footer-link-fontSize); /* 14px */
  line-height: var(--footer-link-lineHeight); /* 2 */
  text-decoration: none;
  transition: color 150ms ease-in-out;
}

.footer-link:hover {
  color: var(--footer-link-colorHover); /* neutral.900 */
}
```

**Navigation-Semantik**

Jede Link-Gruppe sollte in einem `<nav>` Element mit eindeutigem `aria-label` verschachtelt sein:

```
<nav aria-label="Produkt-Links">
  <h3>Produkte</h3>
  <ul>
    <li><a href="/produkte/waschtische">Waschtische</a></li>
    <li><a href="/produkte/duschen">Duschen</a></li>
  </ul>
</nav>
```

**Warum wichtig:** Screen Reader erkennen mehrere Navigations-Landmarks. `aria-label` unterscheidet sie eindeutig.

**Legal-Bereich**

Der Legal-Bereich enthält rechtlich erforderliche Informationen und Links.

**Pflicht-Elemente (Deutschland)****Copyright-Text:**

- Jahreszahl und Firmenname

- Beispiel: © 2024 Hydrophon GmbH

#### Impressum:

- Anbieterkennzeichnung nach § 5 TMG
- Link zu Impressumsseite

#### Datenschutzerklärung:

- Pflicht nach DSGVO Art. 13
- Link zu Datenschutzseite

#### Optional (je nach Website):

- AGB (bei Online-Shop)
- Widerrufsbelehrung (bei Online-Shop)
- Cookie-Einstellungen (DSGVO-Compliance)

### Styling

**Separator:** Links inline mit vertikalem Trennzeichen (|) oder flex mit gap.

**Kleinere Schrift:** 12px für dezente sekundäre Information.

**Dezente Farbe:** neutral.500 für weniger visuelles Gewicht.

**Abstand:** Border und Padding trennen Legal-Bereich vom Hauptfooter.

```
.footer-legal {
  font-size: var(--footer-legal-fontSize); /* 12px */
  color: var(--footer-legal-color); /* neutral.500 */
  margin-top: var(--footer-legal-marginTop); /* 32px */
  border-top: 1px solid var(--footer-legal-borderTop-color); /* neutral.200 */
  padding-top: var(--footer-legal-paddingTop); /* 24px */
}
```

### Layout-Optionen

#### Option 1: Inline mit Separator

```
<div class="footer-legal">
  <p>
    © 2024 Hydrophon GmbH | 
    <a href="/impressum">Impressum</a> | 
    <a href="/datenschutz">Datenschutz</a> | 
    <a href="/agb">AGB</a>
  </p>
</div>
```

#### Option 2: Flex mit Gap (empfohlen)

```
<div class="footer-legal">
  <p>© 2024 Hydrophon GmbH</p>
  <nav aria-label="Rechtliche Links">
    <a href="/impressum">Impressum</a>
    <a href="/datenschutz">Datenschutz</a>
    <a href="/agb">AGB</a>
  </nav>
</div>
```

```
.footer-legal {
  display: flex;
  justify-content: space-between;
  align-items: center;
  gap: var(--spacing-4);
}

.footer-legal nav {
  display: flex;
  gap: var(--spacing-4);
}
```

Vorteil Option 2: Bessere Kontrolle über Layout, keine Text-Separatoren, responsive-freundlicher.

## Tokens

Alle Footer-Styles sind über Design Tokens konfigurierbar:

Token	Wert	CSS-Variable	Beschreibung
footer.background	{neutral.100}	--footer-background	Hintergrundfarbe - dezent vom Hauptinhalt abgesetzt
footer.borderTop.color	{neutral.200}	--footer-borderTop-color	Obere Rahmenlinie
footer.paddingY	{spacing.12} (48px)	--footer-paddingY	Vertikales Padding für großzügigen Abstand
footer.paddingX	{spacing.6} (24px)	--footer-paddingX	Horizontales Padding
footer.maxWidth	1280px	--footer-maxWidth	Maximale Breite konsistent mit Grid
footer.gap	{spacing.8} (32px)	--footer-gap	Abstand zwischen Spalten
footer.heading.fontSize	{fontSize.sm} (14px)	--footer-heading-fontSize	Überschriften-Größe
footer.heading.fontWeight	{fontWeight.semibold} (600)	--footer-heading-fontWeight	Überschriften-Gewicht
footer.heading.color	{neutral.900}	--footer-heading-color	Überschriften-Farbe
footer.heading.textTransform	uppercase	--footer-heading-textTransform	Uppercase für Hierarchie
footer.heading.letterSpacing	0.05em	--footer-heading-letterSpacing	Buchstabenabstand
footer.heading.marginBottom	{spacing.4} (16px)	--footer-heading-marginBottom	Abstand unter Überschrift
footer.link.color	{neutral.600}	--footer-link-color	Link-Farbe
footer.link.colorHover	{neutral.900}	--footer-link-colorHover	Link-Farbe bei Hover
footer.link.fontSize	{fontSize.sm} (14px)	--footer-link-fontSize	Link-Größe
footer.link.lineHeight	2	--footer-link-lineHeight	Touch-Target-freundliche Zeilenhöhe
footer.legal.fontSize	{fontSize.xs} (12px)	--footer-legal-fontSize	Legal-Text Größe
footer.legal.color	{neutral.500}	--footer-legal-color	Legal-Text Farbe
footer.legal.marginTop	{spacing.8} (32px)	--footer-legal-marginTop	Abstand über Legal-Bereich
footer.legal.borderTop.color	{neutral.200}	--footer-legal-borderTop-color	Legal-Bereich obere Linie

footer.legal.paddingTop	{spacing.6} (24px)	--footer-legal-paddingTop	Legal-Bereich oberes Padding
-------------------------	--------------------	---------------------------	------------------------------

**Token-Referenzen:** Siehe `design-system/tokens/navigation.json`

## Code-Beispiele

### HTML - Vollständige Struktur

```

<footer class="footer">
  <div class="footer-container">
    <!-- Optional: Logo -->
    <div class="footer-logo">
      <a href="/">
        
      </a>
    </div>

    <!-- Link-Gruppen -->
    <div class="footer-content">
      <!-- Gruppe 1: Produkte -->
      <nav aria-label="Produkt-Links">
        <h3 class="footer-heading">Produkte</h3>
        <ul>
          <li><a href="/produkte/waschtische">Waschtische</a></li>
          <li><a href="/produkte/duschen">Duschen</a></li>
          <li><a href="/produkte/armaturen">Armaturen</a></li>
          <li><a href="/produkte/zubehoer">Zubehör</a></li>
        </ul>
      </nav>

      <!-- Gruppe 2: Unternehmen -->
      <nav aria-label="Unternehmen">
        <h3 class="footer-heading">Unternehmen</h3>
        <ul>
          <li><a href="/ueber-uns">Über uns</a></li>
          <li><a href="/karriere">Karriere</a></li>
          <li><a href="/nachhaltigkeit">Nachhaltigkeit</a></li>
          <li><a href="/presse">Presse</a></li>
        </ul>
      </nav>

      <!-- Gruppe 3: Service -->
      <nav aria-label="Service">
        <h3 class="footer-heading">Service</h3>
        <ul>
          <li><a href="/support">Support</a></li>
          <li><a href="/downloads">Downloads</a></li>
          <li><a href="/faq">FAQ</a></li>
          <li><a href="/haendlersuche">Händlersuche</a></li>
        </ul>
      </nav>

      <!-- Gruppe 4: Kontakt -->
      <nav aria-label="Kontakt">
        <h3 class="footer-heading">Kontakt</h3>
        <ul>
          <li><a href="/kontakt">Kontaktformular</a></li>
          <li><a href="tel:+4912345678">+49 123 456 78</a></li>
          <li><a href="mailto:info@hydrophon.de">info@hydrophon.de</a></li>
        </ul>
      </nav>
    </div>
  </div>
</footer>

```

```
</nav>
</div>

<!-- Legal-Bereich -->
<div class="footer-legal">
    <p>© 2024 Hydrophon GmbH</p>
    <nav aria-label="Rechtliche Links">
        <a href="/impressum">Impressum</a>
        <a href="/datenschutz">Datenschutz</a>
        <a href="/agb">AGB</a>
    </nav>
</div>
</div>
</footer>
```

## CSS - Vollständiges Styling

```

/* Footer Container */
.footer {
  background-color: var(--footer-background);
  border-top: 1px solid var(--footer-borderTop-color);
  padding: var(--footer-paddingY) var(--footer-paddingX);
}

.footer-container {
  max-width: var(--footer-maxWidth);
  margin: 0 auto;
}

/* Optional: Logo */
.footer-logo {
  margin-bottom: var(--spacing-8);
}

.footer-logo img {
  height: 40px;
  width: auto;
}

/* Link-Gruppen Grid */
.footer-content {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: var(--footer-gap);
  margin-bottom: var(--footer-legal-marginTop);
}

/* Gruppe-Navigation */
.footer-content nav {
  /* Keine zusätzlichen Styles - aria-label ist nicht visuell */
}

/* Überschriften */
.footer-heading {
  font-size: var(--footer-heading-fontSize);
  font-weight: var(--footer-heading-fontWeight);
  color: var(--footer-heading-color);
  text-transform: var(--footer-heading-textTransform);
  letter-spacing: var(--footer-heading-letterSpacing);
  margin-bottom: var(--footer-heading-marginBottom);
}

/* Listen */
.footer-content ul {
  list-style: none;
  padding: 0;
  margin: 0;
}

/* Links */
.footer-content a {
  color: var(--footer-link-color);
}

```

```
font-size: var(--footer-link-fontSize);
line-height: var(--footer-link-lineHeight);
text-decoration: none;
transition: color 150ms ease-in-out;
}

.footer-content a:hover {
  color: var(--footer-link-colorHover);
}

.footer-content a:focus-visible {
  outline: 2px solid var(--hydrophon-blau-300);
  outline-offset: 2px;
  border-radius: 2px;
}

/* Legal-Bereich */
.footer-legal {
  display: flex;
  justify-content: space-between;
  align-items: center;
  gap: var(--spacing-4);
  padding-top: var(--footer-legal-paddingTop);
  border-top: 1px solid var(--footer-legal-borderTop-color);
  font-size: var(--footer-legal-fontSize);
  color: var(--footer-legal-color);
}

.footer-legal p {
  margin: 0;
}

.footer-legal nav {
  display: flex;
  gap: var(--spacing-4);
}

.footer-legal a {
  color: var(--footer-legal-color);
  text-decoration: none;
  transition: color 150ms ease-in-out;
}

.footer-legal a:hover {
  color: var(--neutral-700);
}

/* Responsive: Tablet */
@media (max-width: 1023px) {
  .footer-content {
    grid-template-columns: repeat(2, 1fr);
  }
}

/* Responsive: Mobile */
@media (max-width: 767px) {
  .footer-content {
    grid-template-columns: 1fr;
  }
}
```

```
.footer-legal {  
  flex-direction: column;  
  align-items: flex-start;  
  gap: var(--spacing-3);  
}  
  
.footer-legal nav {  
  flex-direction: column;  
  gap: var(--spacing-2);  
}  
}
```

## React-Beispiel

```

function Footer() {
  const linkGroups = [
    {
      title: "Produkte",
      ariaLabel: "Produkt-Links",
      links: [
        { label: "Waschtische", href: "/produkte/waschtische" },
        { label: "Duschen", href: "/produkte/duschen" },
        { label: "Armaturen", href: "/produkte/armaturen" },
        { label: "Zubehör", href: "/produkte/zubehoer" }
      ]
    },
    {
      title: "Unternehmen",
      ariaLabel: "Unternehmen",
      links: [
        { label: "Über uns", href: "/ueber-uns" },
        { label: "Karriere", href: "/karriere" },
        { label: "Nachhaltigkeit", href: "/nachhaltigkeit" },
        { label: "Presse", href: "/presse" }
      ]
    },
    {
      title: "Service",
      ariaLabel: "Service",
      links: [
        { label: "Support", href: "/support" },
        { label: "Downloads", href: "/downloads" },
        { label: "FAQ", href: "/faq" },
        { label: "Händlersuche", href: "/haendlersuche" }
      ]
    },
    {
      title: "Kontakt",
      ariaLabel: "Kontakt",
      links: [
        { label: "Kontaktformular", href: "/kontakt" },
        { label: "+49 123 456 78", href: "tel:+4912345678" },
        { label: "info@hydrophon.de", href: "mailto:info@hydrophon.de" }
      ]
    }
  ];
}

const legalLinks = [
  { label: "Impressum", href: "/impressum" },
  { label: "Datenschutz", href: "/datenschutz" },
  { label: "AGB", href: "/agb" }
];

return (
  <footer className="footer">
    <div className="footer-container">
      <div className="footer-content">
        {linkGroups.map((group) => (
          <nav key={group.ariaLabel} aria-label={group.ariaLabel}>

```

```

        <h3 className="footer-heading">{group.title}</h3>
        <ul>
          {group.links.map((link) => (
            <li key={link.href}>
              <a href={link.href}>{link.label}</a>
            </li>
          ))}
        </ul>
      </nav>
    )}
</div>

<div className="footer-legal">
  <p>© {new Date().getFullYear()} Hydrophon GmbH</p>
  <nav aria-label="Rechtliche Links">
    {legalLinks.map((link) => (
      <a key={link.href} href={link.href}>
        {link.label}
      </a>
    ))}
  </nav>
</div>
</div>
</footer>
);
}

```

## Barrierefreiheit

Der Footer muss für alle Nutzer zugänglich sein, unabhängig von ihren Fähigkeiten oder verwendeten Technologien.

### Landmark: role="contentinfo"

**Automatisch:** Das `<footer>` Element hat automatisch `role="contentinfo"`.

**Screen Reader-Ansage:** "Contentinfo Landmark" oder "Footer Landmark".

**Warum wichtig:** Screen Reader-Nutzer können direkt zum Footer springen (Tastenkürzel in NVDA/JAWS).

**Einschränkung:** Nur ein `contentinfo` Landmark pro Seite (äußerstes `<footer>` Element).

### Mehrere nav-Elemente

**Problem:** Footer enthält mehrere `<nav>` Elemente. Screen Reader kündigen alle als "Navigation" an.

**Lösung:** Jedes `<nav>` benötigt eindeutiges `aria-label`.

```

<nav aria-label="Produkt-Links">...</nav>
<nav aria-label="Unternehmen">...</nav>
<nav aria-label="Service">...</nav>
<nav aria-label="Rechtliche Links">...</nav>

```

**Screen Reader-Ansage:** "Produkt-Links Navigation", "Unternehmen Navigation", etc.

## Skip-Links

**Problem:** Keyboard-Nutzer müssen alle Header- und Footer-Links durchtabben.

**Lösung:** Skip-Link am Seitenanfang ("Zum Hauptinhalt springen").

```
<a href="#main" class="skip-link">Zum Hauptinhalt springen</a>

<header>...</header>

<main id="main">...</main>

<footer>...</footer>
```

**Styling:** Skip-Link visuell versteckt, sichtbar bei Fokus.

```
.skip-link {
  position: absolute;
  top: -40px;
  left: 0;
  z-index: 100;
}

.skip-link:focus {
  top: 0;
}
```

## Touch-Targets

**WCAG 2.5.5 (AAA):** Touch-Targets sollten mindestens 44×44px sein.

**Erfüllung:** Footer-Links haben `line-height: 2`, was bei 14px Schrift 28px ergibt. Für AAA-Compliance zusätzliches Padding hinzufügen:

```
.footer-content a {
  display: inline-block;
  padding: 8px 0; /* Erreicht 44px Höhe: 8 + 28 + 8 */
}
```

## Kontrast

**WCAG 1.4.3:** Text benötigt 4.5:1 Kontrast (oder 3:1 für große Schrift  $\geq 18\text{px}$ ).

**Erfüllung:**

- Footer-Links: `neutral.600 (#757575)` auf `neutral.100 (#f5f5f5)` = 4.5:1 ✓
- Legal-Text: `neutral.500 (#9e9e9e)` auf `neutral.100` = 3.2:1 (grenzwertig)

**Empfehlung bei strenger WCAG-Compliance:** Legal-Text auf `neutral.600` anpassen.

## Focus-Indicator

**Standard:** Konsistent mit globalen Link-Styles (2px outline, 2px offset).

**WCAG 2.4.7:** Focus muss visuell erkennbar sein.

**Erfüllung:** Siehe CSS-Beispiele - `hydrophon.blau.300 outline`.

## Best Practices

### Konsistenz

**Position:** Footer sollte auf allen Seiten an derselben Position erscheinen.

**Inhalt:** Gleiche Link-Gruppen und Legal-Links auf allen Seiten (außer spezielle Landing Pages).

**Styling:** Einheitliche Farben, Abstände, Typografie.

### Klare Gruppierung

**Maximum 4-5 Gruppen:** Mehr verwirrt Nutzer.

**Logische Themen:** Verwandte Links gruppieren (nicht alphabetisch).

**Überschriften:** Kurz und prägnant (1-2 Wörter).

### Legal immer unten

**Trennung:** Visuell vom Hauptfooter getrennt (Border, zusätzlicher Abstand).

**Vollständigkeit:** Alle rechtlich erforderlichen Links enthalten.

**Aktualität:** Copyright-Jahr automatisch aktualisieren (JavaScript oder Server-Side).

### Performance

**Bilder optimieren:** Logo als SVG oder optimiertes PNG/WebP.

**Lazy Loading:** Nicht erforderlich für Footer (unterhalb Fold).

**Critical CSS:** Footer-Styles können nachgeladen werden.

## Nicht verwenden

### Zu viele Links ohne Gruppierung

**Falsch:** 20+ Links ohne thematische Gruppierung.

**Problem:** Überfordernd, schwer zu scannen, keine Orientierung.

**Richtig:** Maximum 4-5 Gruppen mit je 4-6 Links.

### Fehlende Landmark-Region

**Falsch:**

```
<div class="footer">...</div>
```

**Problem:** Screen Reader erkennen keine contentinfo-Landmark.

**Richtig:**

```
<footer class="footer">...</footer>
```

## nav ohne aria-label

Falsch:

```
<nav>
  <h3>Produkte</h3>
  ...
</nav>
<nav>
  <h3>Service</h3>
  ...
</nav>
```

**Problem:** Screen Reader kündigen beide als "Navigation" an - nicht unterscheidbar.

**Richtig:** Siehe Code-Beispiele mit `aria-label`.

## Copyright ohne Jahr

Falsch: © Hydrophon GmbH (statisch)

Problem: Wirkt veraltet, rechtlich unklar.

Richtig: © 2024 Hydrophon GmbH (automatisch aktualisiert)

```
<p>© {new Date().getFullYear()} Hydrophon GmbH</p>
```

## Referenzen

**Token-System:** Siehe `design-system/tokens/navigation.json` für alle Footer-Tokens.

**Header-Navigation:** Konsistente Focus-Indicator und Link-Styles.

**Grid-System:** Footer-Breite konsistent mit Grid-Container (1280px).

**WCAG 2.1 AA:** [Understanding WCAG 2.1](#)

**Contentinfo Landmark:** [W3C ARIA Landmarks](#)

# Product Card

---

## Übersicht

Product Cards präsentieren Hydrophon-Produkte in scannable Grid-Listen. Sie zeigen die wichtigsten Produktinformationen auf einen Blick und ermöglichen schnellen Zugriff auf Produktdetails.

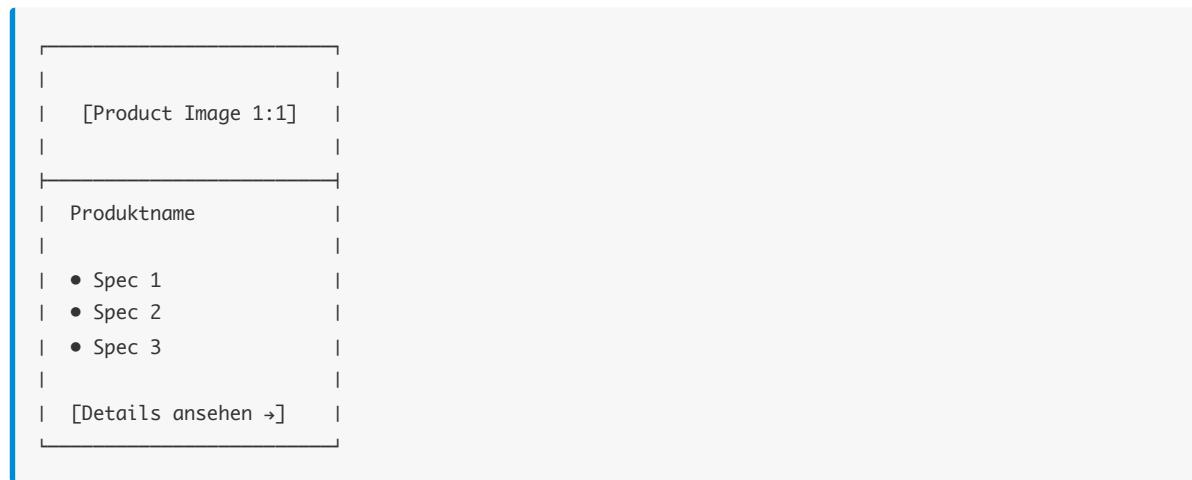
**Verwendung:** Produktkataloge, Kategorie-Seiten, verwandte Produkte, Suchergebnisse

**Kontext:** B2B-Nutzer möchten Produkte schnell vergleichen und technische Spezifikationen erfassen. Die Card-Darstellung ermöglicht effizientes Scannen großer Produktlisten.

## Anatomie

Eine Product Card besteht aus folgenden Elementen:

1. **Card Container** ( `<article>` ) – Semantischer Container für Produktinformation
2. **Product Image** – Quadratisches Produktbild (1:1 Aspect Ratio)
3. **Content Area** – Text-Container mit Padding
4. **Title** ( `<h3>` ) – Produktnname als Heading
5. **Specs List** ( `<ul>` ) – 2-3 technische Spezifikationen
6. **CTA Link** – "Details ansehen" oder ähnliche Aktion



## Grid-Layout

### CSS Grid auto-fit Pattern

Product Cards nutzen ein modernes CSS Grid-Pattern, das ohne Media Queries responsive Layouts ermöglicht:

```
.product-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(min(280px, 100%), 1fr));
  gap: var(--card-gap); /* 24px */
}
```

#### Wie funktioniert dieses Pattern?

- repeat(auto-fit, ...) – Grid erstellt automatisch so viele Spalten wie möglich
- minmax(min(280px, 100%), 1fr) – Jede Spalte ist mindestens 280px breit (oder 100% auf sehr kleinen Screens)
- 1fr – Spalten teilen verfügbaren Platz gleichmäßig
- auto-fit kollabiert leere Spalten, sodass vorhandene Cards mehr Platz bekommen

#### Vorteile:

- Keine Breakpoints in JavaScript oder CSS erforderlich
- Cards passen sich automatisch an Container-Breite an
- Funktioniert in jeder Container-Größe (nicht nur Viewport)
- Intrinsic Responsive Design

## Responsive Verhalten

Das Grid passt sich automatisch an:

Viewport-Breite	Spalten	Card-Breite
< 560px	1	100%
560-840px	2	~260px
840-1120px	3	~266px
> 1120px	4	~280px

Diese Breakpoints entstehen **automatisch** durch die minmax(280px, 1fr) Berechnung. Bei größeren Viewports können auch 5+ Spalten entstehen.

## Hover-Effekt

Product Cards reagieren auf Hover mit einer subtilen Animation, die Interaktivität signalisiert:

#### Effekt:

- Card hebt sich um 4px ( translateY(-4px) )
- Shadow wechselt von sm zu md
- Transition dauert 300ms mit ease Timing

#### Warum Transform statt Position?

- transform: translateY() ist GPU-beschleunigt
- Höhere Performance als top oder margin
- Löst kein Layout-Reflow aus

#### Accessibility:

- prefers-reduced-motion: reduce deaktiviert Animation für Nutzer mit Vestibular Disorders

```
.product-card {
  transition: transform 300ms ease, box-shadow 300ms ease;
}

.product-card:hover {
  transform: translateY(-4px);
  box-shadow: var(--card-shadow-hover);
}

@media (prefers-reduced-motion: reduce) {
  .product-card {
    transition: none;
  }
  .product-card:hover {
    transform: none;
  }
}
```

## Performance-Optimierung

### Shadow-Animation mit Pseudo-Element

**Problem:** `box-shadow` direkt zu animieren ist CPU-intensiv und kann zu Jank führen.

**Lösung:** Nutze ein `::after` Pseudo-Element mit `opacity`:

```
.product-card {
  position: relative;
  box-shadow: var(--card-shadow-default);
}

.product-card::after {
  content: '';
  position: absolute;
  inset: 0;
  border-radius: inherit;
  box-shadow: var(--card-shadow-hover);
  opacity: 0;
  transition: opacity 300ms ease;
  pointer-events: none;
  z-index: -1;
}

.product-card:hover::after {
  opacity: 1;
}
```

**Vorteil:** Nur `opacity` wird animiert (GPU-beschleunigt), Shadow ist pre-rendered.

### Lazy Loading

Product Images sollten lazy-loaded werden, um Initial Page Load zu beschleunigen:

```

```

- `loading="lazy"` – Browser lädt Bild erst kurz bevor es ins Viewport scrollt
- `decoding="async"` – Bild-Dekodierung blockiert nicht Main Thread

## will-change Sparsam Nutzen

```
/* NICHT auf alle Cards anwenden! */  
.product-card:hover,  
.product-card:focus-within {  
  will-change: transform;  
}
```

`will-change: transform` bereitet GPU-Layers vor, verbraucht aber Speicher. Nur auf Hover/Focus-Kandidaten anwenden, nicht auf alle Cards gleichzeitig.

## Tokens

Product Cards nutzen Tokens aus `tokens/cards.json` :

Token	Wert	Verwendung
card.minWidth	280px	Grid minmax() Berechnung
card.gap	24px (spacing.6)	Abstand zwischen Cards
card.padding	24px (spacing.6)	Innenabstand
card.radius	8px (borderRadius.lg)	Abgerundete Ecken
card.border.width	1px	Border-Stärke
card.border.color	neutral.200	Border-Farbe
card.shadow.default	shadow.sm	Shadow im Ruhezustand
card.shadow.hover	shadow.md	Shadow bei Hover
card.transition.duration	300ms	Animationsdauer
card.hover.transform	translateY(-4px)	Hover-Bewegung
card.image.aspectRatio	1 / 1	Quadratisches Bild
card.image.background	neutral.100	Placeholder-Hintergrund
card.title.fontSize	18px (fontSize.lg)	Titel-Größe
card.title.fontWeight	600 (semibold)	Titel-Gewicht
card.specs.fontSize	14px (fontSize.sm)	Specs-Größe
card.specs.color	neutral.700	Specs-Farbe
card.specs.gap	4px (spacing.1)	Abstand zwischen Specs
card.cta.color	hydrophon.blau.600	CTA-Linkfarbe
card.cta.colorHover	hydrophon.blau.700	CTA-Linkfarbe Hover

## Code-Beispiele

### HTML-Struktur

```
<div class="product-grid">

  <article class="product-card">
    <a href="/produkte/ablauftrinne-80cm" class="product-card__link">
      <div class="product-card__image">
        
      </div>
      <div class="product-card__content">
        <h3 class="product-card__title">Ablaufrinne 80cm</h3>
        <ul class="product-card__specs">
          <li>Länge: 80cm</li>
          <li>Material: Edelstahl</li>
          <li>Ablauf: Seitlich</li>
        </ul>
        <span class="product-card__cta">
          Details ansehen →
        </span>
      </div>
    </a>
  </article>

  <article class="product-card">
    <!-- Weitere Cards... -->
  </article>

</div>
```

#### Semantik:

- `<article>` für jede Card (self-contained content)
- `<a>` wrappt gesamte Card (gesamte Card klickbar)
- `<h3>` für Produktnamen (wenn Cards in Sektion mit `<h2>` Überschrift)
- `<ul>` für Specs-Liste (semantische Liste)

## CSS-Implementierung

```
/* Grid Container */
.product-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(min(280px, 100%), 1fr));
  gap: var(--card-gap);
}

/* Card Container */
.product-card {
  position: relative;
  background: var(--card-background);
  border: var(--card-border-width) solid var(--card-border-color);
  border-radius: var(--card-radius);
  box-shadow: var(--card-shadow-default);
  overflow: hidden;
  transition: transform var(--card-transition-duration) var(--card-transition-timing),
              box-shadow var(--card-transition-duration) var(--card-transition-timing);
}

.product-card:hover {
  transform: var(--card-hover-transform);
  box-shadow: var(--card-shadow-hover);
}

/* Link (entire card clickable) */
.product-card__link {
  display: block;
  text-decoration: none;
  color: inherit;
}

/* Image Container */
.product-card__image {
  aspect-ratio: var(--card-image-aspect-ratio);
  background: var(--card-image-background);
  overflow: hidden;
}

.product-card__image img {
  width: 100%;
  height: 100%;
  object-fit: var(--card-image-object-fit);
}

/* Content Area */
.product-card__content {
  padding: var(--card-padding);
  display: flex;
  flex-direction: column;
  gap: var(--spacing-3);
}

/* Title */
.product-card__title {
  margin: 0;
```

```
font-size: var(--card-title-font-size);
font-weight: var(--card-title-font-weight);
color: var(--card-title-color);
}

/* Specs List */
.product-card__specs {
  margin: 0;
  padding: 0;
  list-style: none;
  font-size: var(--card-specs-font-size);
  color: var(--card-specs-color);
  line-height: var(--card-specs-line-height);
  display: flex;
  flex-direction: column;
  gap: var(--card-specs-gap);
}

.product-card__specs li::before {
  content: '●';
  margin-right: 8px;
  color: var(--neutral-400);
}

/* CTA */
.product-card__cta {
  display: inline-flex;
  align-items: center;
  gap: 4px;
  margin-top: auto; /* Push to bottom */
  font-weight: var(--card-cta-font-weight);
  color: var(--card-cta-color);
  transition: color 150ms ease;
}

.product-card:hover .product-card__cta {
  color: var(--card-cta-color-hover);
}

/* Reduced Motion */
@media (prefers-reduced-motion: reduce) {
  .product-card {
    transition: none;
  }
  .product-card:hover {
    transform: none;
  }
}
```

## Performance-optimierte Shadow-Animation

```
.product-card {
  position: relative;
  box-shadow: var(--card-shadow-default);
  /* ... andere Styles ... */
}

/* Pseudo-Element für Hover-Shadow */
.product-card::after {
  content: '';
  position: absolute;
  inset: 0;
  border-radius: inherit;
  box-shadow: var(--card-shadow-hover);
  opacity: 0;
  transition: opacity 300ms ease;
  pointer-events: none;
  z-index: -1;
}

.product-card:hover::after {
  opacity: 1;
}
```

## Barrierefreiheit

### Semantisches HTML

- `<article>` für Card-Container (self-contained content)
- `<h3>` für Produktnamen (korrekte Heading-Hierarchie)
- `<ul>` für Specs-Liste (semantische Struktur)

### Keyboard-Navigation

- Gesamte Card als `<a>` wrappt ermöglicht Fokus auf gesamter Card
- `:focus-visible` State für Keyboard-Navigation (ohne Hover)
- Enter oder Space aktiviert Link

```
.product-card:focus-within {
  outline: 2px solid var(--hydrophon-blau-500);
  outline-offset: 2px;
}
```

### Screen Reader

- `alt`-Attribut auf Images mit beschreibendem Text: "Hydrophon Ablaufrinne 80cm"
- Heading (`<h3>`) wird als Produktnamen erkannt
- Specs-Liste wird als Liste vorgelesen

## Lazy Loading

- `loading="lazy"` ist accessibility-friendly (kein JavaScript erforderlich)
- Screen Reader lesen `alt`-Text unabhängig vom Loading-Status

## Best Practices

### Dos

- CSS Grid auto-fit nutzen – Responsive ohne Media Queries
- Transform für Animationen – GPU-beschleunigt, bessere Performance
- Lazy Loading für Images – Schnellerer Initial Load
- Semantisches HTML – `<article>`, `<h3>`, `<ul>` für bessere Accessibility
- `prefers-reduced-motion` – Animationen deaktivieren für Motion-Sensitive Users
- Alt-Text auf Images – Beschreibend und informativ
- 2-3 Specs maximal – Übersichtlich, nicht überladen

### Don'ts

- Feste Spaltenanzahl vermeiden – `grid-template-columns: repeat(4, 1fr)` ist nicht responsive
- box-shadow direkt animieren – CPU-intensiv, nutze Pseudo-Element mit `opacity`
- will-change auf allen Cards – Speicher-Verschwendungen, nur auf Hover
- Zu viele Specs – Mehr als 3 überladen die Card
- Bilder ohne alt-Attribut – Screen Reader können Kontext nicht vermitteln
- Click-Handler auf Card-Container – Nutze `<a>` für native Keyboard-Unterstützung

## Varianten

### Ohne CTA-Link

Falls gesamte Card bereits Link ist, kann separater CTA weggelassen werden:

```
<article class="product-card">
  <a href="/produkte/ablaufrinne-80cm">
    <div class="product-card__image">...</div>
    <div class="product-card__content">
      <h3 class="product-card__title">Ablaufrinne 80cm</h3>
      <ul class="product-card__specs">...</ul>
    </div>
  </a>
</article>
```

### Mit Badge

Für "Neu" oder "Sale" Badges:

```
<div class="product-card__image">
  <span class="product-card__badge">Neu</span>
  
</div>
```

```
.product-card__badge {
  position: absolute;
  top: 12px;
  right: 12px;
  padding: 4px 12px;
  background: var(--hydrophon-blau-500);
  color: white;
  font-size: var(--font-size-xs);
  font-weight: var(--font-weight-semibold);
  border-radius: var(--border-radius-base);
  z-index: 1;
}
```

## Mit Preis

```
<div class="product-card__content">
  <h3 class="product-card__title">Ablaufrinne 80cm</h3>
  <p class="product-card__price">189,00 €</p>
  <ul class="product-card__specs">...</ul>
</div>
```

```
.product-card__price {
  font-size: var(--font-size-xl);
  font-weight: var(--font-weight-bold);
  color: var(--hydrophon-blau-600);
  margin: 0;
}
```

## Beispiel-Integration

```
<section class="product-catalog">
  <h2>Unsere Ablaufrinnen</h2>

  <div class="product-grid">
    <article class="product-card">...</article>
    <article class="product-card">...</article>
    <article class="product-card">...</article>
    <article class="product-card">...</article>
    <article class="product-card">...</article>
    <article class="product-card">...</article>
  </div>
</section>
```

## Verwandte Komponenten

- Content Card – Allgemeine Inhalte (Blog, Features)
- Data Table – Produkt-Vergleichstabellen
- Button – CTA-Buttons in Cards

## Ressourcen

- CSS Grid auto-fit vs auto-fill
- Lazy Loading Images
- prefers-reduced-motion
- WCAG 2.1 - Focus Visible

# Content Card

---

## Übersicht

Content Cards präsentieren allgemeine Inhalte wie Blog-Posts, News-Artikel, Features oder Service-Angebote. Im Gegensatz zu Product Cards sind sie flexibler in Struktur und Inhalt.

**Verwendung:** Blog-Übersichten, News-Bereiche, Feature-Highlights, Service-Kacheln, Team-Profile

**Kontext:** Content Cards ermöglichen visuelle Hierarchie für verschiedene Content-Typen. Sie sind anpassbar und können mit oder ohne Bild, mit Meta-Informationen und unterschiedlichen Call-to-Actions verwendet werden.

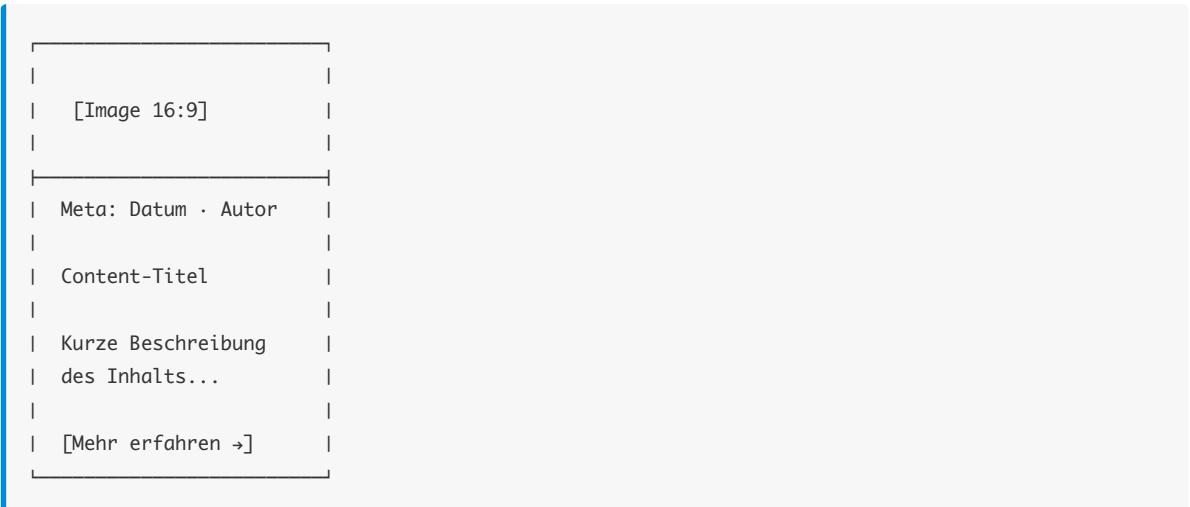
## Unterschied zu Product Cards

Product Card	Content Card
Quadratisches Bild (1:1)	Flexibles Bildformat (16:9, 4:3, oder kein Bild)
Specs-Liste (immer)	Optional: Description, Excerpt
Produkt-CTA	Flexibler CTA-Text
Grid-optimiert	Grid oder List-Layout

## Anatomie

Eine Content Card besteht aus folgenden Elementen:

1. **Card Container** ( `<article>` ) – Semantischer Container
2. **Image** (optional) – Bild mit flexiblem Aspect Ratio
3. **Content Area** – Text-Container
4. **Meta** (optional) – Datum, Autor, Kategorie
5. **Title** ( `<h3>` ) – Content-Überschrift
6. **Description/Excerpt** (optional) – Kurzbeschreibung
7. **CTA Link** (optional) – "Mehr erfahren", "Artikel lesen"



## Varianten

### 1. Vertikale Card mit Bild

Standard-Variante: Bild oben, Content unten.

```
<article class="content-card content-card--vertical">
  <div class="content-card__image">
    
  </div>
  <div class="content-card__content">
    <div class="content-card__meta">
      <span class="content-card__date">15. Januar 2026</span>
      <span class="content-card__category">Installationstipps</span>
    </div>
    <h3 class="content-card__title">Professionelle Installation von Bodenabläufen</h3>
    <p class="content-card__description">
      Erfahren Sie, wie Sie Hydrophon Bodenabläufe fachgerecht installieren
      und häufige Fehler vermeiden.
    </p>
    <a href="/blog/installation-bodenablaeufe" class="content-card__cta">
      Artikel lesen →
    </a>
  </div>
</article>
```

### 2. Horizontale Card mit Bild

Bild links, Content rechts – für breitere Layouts.

```
<article class="content-card content-card--horizontal">
  <div class="content-card__image">
    
  </div>
  <div class="content-card__content">
    <div class="content-card__meta">
      <span class="content-card__date">12. Januar 2026</span>
    </div>
    <h3 class="content-card__title">Neue Produktlinie hyIndustry</h3>
    <p class="content-card__description">
      Hydrophon erweitert Portfolio um industrielle Ablaufsysteme.
    </p>
    <a href="/news/hyindustry-launch" class="content-card__cta">
      Mehr erfahren →
    </a>
  </div>
</article>
```

CSS für horizontales Layout:

```
.content-card--horizontal {
  display: grid;
  grid-template-columns: 280px 1fr;
  gap: var(--spacing-6);
}

@media (max-width: 768px) {
  .content-card--horizontal {
    grid-template-columns: 1fr;
  }
}
```

### 3. Card ohne Bild (Text-Only)

Für textbasierte Inhalte oder wenn kein passendes Bild verfügbar ist.

```
<article class="content-card content-card--text-only">
  <div class="content-card__content">
    <div class="content-card__meta">
      <span class="content-card__category">Service</span>
    </div>
    <h3 class="content-card__title">24/7 Technischer Support</h3>
    <p class="content-card__description">
      Unser Support-Team steht Ihnen bei Fragen zu Installation,
      Wartung und Produktauswahl zur Verfügung.
    </p>
    <a href="/service/support" class="content-card__cta">
      Kontakt aufnehmen →
    </a>
  </div>
</article>
```

## 4. Feature Card mit Icon

Für Service-Features oder Produkt-Highlights.

```
<article class="content-card content-card--feature">
  <div class="content-card__content">
    <div class="content-card__icon">
      <svg><!-- Lucide Icon --></svg>
    </div>
    <h3 class="content-card__title">Schnelle Lieferung</h3>
    <p class="content-card__description">
      Alle Produkte ab Lager verfügbar. Lieferung innerhalb von 2-3 Werktagen.
    </p>
  </div>
</article>
```

CSS für Icon:

```
.content-card__icon {
  width: 48px;
  height: 48px;
  display: flex;
  align-items: center;
  justify-content: center;
  background: var(--hydrophon-blau-50);
  border-radius: var(--border-radius-lg);
  margin-bottom: var(--spacing-4);
}

.content-card__icon svg {
  width: 24px;
  height: 24px;
  color: var(--hydrophon-blau-600);
}
```

## States

### Default

Basis-State ohne Interaktion.

### Hover

Gleicher Effekt wie Product Cards:

```
.content-card:hover {
  transform: translateY(-4px);
  box-shadow: var(--card-shadow-hover);
}
```

## Focus

Für Keyboard-Navigation:

```
.content-card:focus-within {
  outline: 2px solid var(--hydrophon-blau-500);
  outline-offset: 2px;
}
```

## Disabled (optional)

Falls Content nicht verfügbar oder archiviert:

```
.content-card--disabled {
  opacity: 0.6;
  pointer-events: none;
}
```

```
<article class="content-card content-card--disabled" aria-disabled="true">
  <!-- ... -->
</article>
```

## Tokens

Content Cards nutzen die gleichen Tokens wie Product Cards aus `tokens/cards.json`:

Token	Wert	Verwendung
card.padding	24px	Innenabstand
card.radius	8px	Abgerundete Ecken
card.shadow.default	shadow.sm	Shadow im Ruhezustand
card.shadow.hover	shadow.md	Shadow bei Hover
card.transition.duration	300ms	Animationsdauer
card.title.fontSize	18px	Titel-Größe
card.title.fontWeight	600	Titel-Gewicht
card.cta.color	hydrophon.blau.600	CTA-Farbe

Zusätzliche Token für Content Cards:

- `fontSize.sm` (14px) – Meta-Informationen
- `color.text.secondary` (neutral.700) – Description-Text
- `color.text.muted` (neutral.500) – Meta-Text

## Code-Beispiele

### Vertikale Card (vollständig)

```
<article class="content-card content-card--vertical">
  <a href="/blog/installation-tipps" class="content-card__link">
    <div class="content-card__image">
      
    </div>
    <div class="content-card__content">
      <div class="content-card__meta">
        <time datetime="2026-01-15" class="content-card__date">15. Januar 2026</time>
        <span class="content-card__category">Installationstipps</span>
      </div>
      <h3 class="content-card__title">Professionelle Installation von Bodenabläufen</h3>
      <p class="content-card__description">
        Erfahren Sie, wie Sie Hydrophon Bodenabläufe fachgerecht installieren
        und häufige Fehler vermeiden. Mit praktischen Schritt-für-Schritt-Anleitungen.
      </p>
      <span class="content-card__cta">
        Artikel lesen →
      </span>
    </div>
  </a>
</article>
```

## CSS-Implementierung

```
/* Card Container */
.content-card {
  position: relative;
  background: var(--card-background);
  border: var(--card-border-width) solid var(--card-border-color);
  border-radius: var(--card-radius);
  box-shadow: var(--card-shadow-default);
  overflow: hidden;
  transition: transform var(--card-transition-duration) ease,
              box-shadow var(--card-transition-duration) ease;
}

.content-card:hover {
  transform: translateY(-4px);
  box-shadow: var(--card-shadow-hover);
}

/* Link */
.content-card__link {
  display: block;
  text-decoration: none;
  color: inherit;
}

/* Image */
.content-card__image {
  aspect-ratio: 16 / 9;
  background: var(--card-image-background);
  overflow: hidden;
}

.content-card__image img {
  width: 100%;
  height: 100%;
  object-fit: cover;
}

/* Content Area */
.content-card__content {
  padding: var(--card-padding);
  display: flex;
  flex-direction: column;
  gap: var(--spacing-3);
}

/* Meta */
.content-card__meta {
  display: flex;
  align-items: center;
  gap: var(--spacing-3);
  font-size: var(--font-size-sm);
  color: var(--color-text-muted);
}

.content-card__date::after {
```

```
content: '.';
margin-left: var(--spacing-3);
}

.content-card__category {
  text-transform: uppercase;
  letter-spacing: 0.05em;
  font-weight: var(--font-weight-medium);
  color: var(--hydrophon-blau-600);
}

/* Title */
.content-card__title {
  margin: 0;
  font-size: var(--card-title-font-size);
  font-weight: var(--card-title-font-weight);
  color: var(--card-title-color);
  line-height: 1.3;
}

/* Description */
.content-card__description {
  margin: 0;
  font-size: var(--font-size-base);
  color: var(--color-text-secondary);
  line-height: 1.6;
  display: -webkit-box;
  -webkit-box-orient: vertical;
  -webkit-line-clamp: 3;
  overflow: hidden;
}

/* CTA */
.content-card__cta {
  display: inline-flex;
  align-items: center;
  gap: 4px;
  margin-top: auto;
  font-weight: var(--card-cta-font-weight);
  color: var(--card-cta-color);
  transition: color 150ms ease;
}

.content-card:hover .content-card__cta {
  color: var(--card-cta-color-hover);
}

/* Horizontal Variant */
.content-card--horizontal {
  display: grid;
  grid-template-columns: 280px 1fr;
  gap: var(--spacing-6);
}

.content-card--horizontal .content-card__image {
  aspect-ratio: 4 / 3;
}

@media (max-width: 768px) {
```

```

.content-card--horizontal {
  grid-template-columns: 1fr;
}

/* Text-Only Variant */
.content-card--text-only .content-card__content {
  padding: var(--spacing-8);
}

/* Feature Card Variant */
.content-card--feature {
  text-align: center;
  border: 2px dashed var(--neutral-200);
}

.content-card--feature:hover {
  border-color: var(--hydrophon-blau-300);
}

/* Disabled State */
.content-card--disabled {
  opacity: 0.6;
  pointer-events: none;
}

/* Reduced Motion */
@media (prefers-reduced-motion: reduce) {
  .content-card {
    transition: none;
  }
  .content-card:hover {
    transform: none;
  }
}

```

## Barrierefreiheit

### Semantisches HTML

- `<article>` für Card (self-contained content)
- `<h3>` für Titel (korrekte Heading-Hierarchie)
- `<time datetime="...">` für Datum (maschinenlesbar)
- `<a>` für klickbare Card (native Keyboard-Unterstützung)

### Screen Reader

- `alt` -Attribut auf Images beschreibt Bildinhalt
- `datetime` -Attribut auf `<time>` für maschinenlesbare Datumsformate
- Heading-Hierarchie ermöglicht Navigation via Headings

## Keyboard-Navigation

```
.content-card:focus-within {
  outline: 2px solid var(--hydrophon-blau-500);
  outline-offset: 2px;
}
```

## ARIA für Disabled State

```
<article class="content-card content-card--disabled" aria-disabled="true">
  <!-- ... -->
</article>
```

## Best Practices

### Dos

- **Flexibles Image Aspect Ratio** – 16:9 für Landscape, 4:3 für Square, 1:1 für Portrait
- **Line-Clamp für Description** – Begrenzt Text auf 3 Zeilen, verhindert Layout-Bruch
- **Meta-Informationen gruppieren** – Datum, Autor, Kategorie logisch zusammenfassen
- **CTA-Text anpassen** – "Artikel lesen", "Mehr erfahren", "Details ansehen" je nach Kontext
- **Semantic HTML** – `<article>`, `<time>`, `<h3>` für bessere Accessibility

### Don'ts

- **Zu langer Description-Text** – Nutze `line-clamp` oder `max-height` mit `overflow: hidden`
- **Fehlende Meta-Informationen** – Nutzer erwarten Datum/Kategorie bei News/Blog
- **Generischer CTA** – "Mehr" oder "Klick hier" sind nicht aussagekräftig
- **Zu viele Varianten mischen** – Einheitlichkeit innerhalb einer Sektion

## Grid-Layout für Content Cards

Gleicher `auto-fit` Ansatz wie bei Product Cards:

```
.content-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(min(320px, 100%), 1fr));
  gap: var(--card-gap);
}
```

**Hinweis:** Content Cards können breiter sein als Product Cards (320px statt 280px), da oft mehr Text enthalten ist.

## Verwandte Komponenten

- **Product Card** – Produkt-spezifische Cards

- **Data Table** – Tabellarische Darstellung
- **Button** – CTA-Buttons in Cards

## Ressourcen

- [HTML5 `<article>` Element](#)
  - [CSS `line-clamp`](#)
  - [WCAG 2.1 - Time Element](#)
-

# Data Table

---

## Übersicht

Data Tables präsentieren strukturierte Daten in tabellarischer Form. Im B2B-Kontext von Hydrophon werden sie primär für Produktvergleiche und technische Spezifikationen verwendet.

**Verwendung:** Produktvergleiche, technische Datenblätter, Preislisten, Spezifikationstabellen

**Kontext:** B2B-Nutzer müssen Produkte anhand technischer Kriterien vergleichen (Länge, Breite, Material, Ablaufleistung, Preis). Tables ermöglichen schnelles Scannen und Vergleichen.

## Anatomie

Eine Data Table besteht aus folgenden Elementen:

1. `<table>` – Semantischer Container
2. `<caption>` – Tabellenbeschreibung (für Screen Reader wichtig)
3. `<thead>` – Header-Bereich mit Spaltenüberschriften
4. `<th scope="col">` – Spalten-Header
5. `<tbody>` – Daten-Bereich
6. `<th scope="row">` – Zeilen-Header (erste Spalte, z.B. Produktnname)
7. `<td>` – Daten-Zellen
8. `<tfoot>` (optional) – Footer-Bereich für Summen/Durchschnitt

Tabellentitel (caption)		
Spalte 1 (th)	Spalte 2 (th)	Spalte 3 (th)
Zeile 1 (th)	Daten (td)	Daten (td)
Zeile 2	Daten (td)	Daten (td)
Zeile 3	Daten (td)	Daten (td)

## Header-Struktur

### Spalten-Header (`thead`)

Alle Header-Zellen in `<thead>` benötigen `scope="col"` :

```
<thead>
  <tr>
    <th scope="col">Produkt</th>
    <th scope="col">Länge</th>
    <th scope="col">Breite</th>
    <th scope="col">Material</th>
    <th scope="col">Preis</th>
  </tr>
</thead>
```

Warum `scope="col"` ?

- Screen Reader ordnen Datenzellen dem korrekten Header zu
- WCAG 1.3.1 - Info and Relationships
- Ermöglicht Navigation via Screen Reader Table Commands

### Zeilen-Header (erste Spalte)

Erste Spalte in `<tbody>` sollte `<th scope="row">` nutzen:

```
<tbody>
  <tr>
    <th scope="row">Ablaufrinne 80cm</th>
    <td>80cm</td>
    <td>10cm</td>
    <td>Edelstahl</td>
    <td>189,00 €</td>
  </tr>
</tbody>
```

Warum `scope="row"` ?

- Produktname ist Zeilen-Identifikator
- Screen Reader lesen: "Ablaufrinne 80cm, Länge: 80cm"
- Bessere Navigation für blinde Nutzer

## Zebra-Striping

### Subtiler Kontrast

Zebra-Striping verbessert Lesbarkeit durch alternierende Zeilenhintergründe:

```
tbody tr:nth-child(odd) {
  background: transparent;
}

tbody tr:nth-child(even) {
  background: var(--table-row-stripe); /* neutral.50 */
}
```

### Warum subtil?

- Starke Kontraste (z.B. neutral.100 vs weiß) = visuelle Störung
- Subtile Kontraste (neutral.50 vs weiß) = verbesserte Scan-Geschwindigkeit
- Kontrast zwischen Zeilen: ~1.5:1 (ausreichend für Orientierung)
- Text-Kontrast auf beiden Hintergründen: 4.5:1 (WCAG AA compliant)

### Kontrast-Kalkulation:

- Weiß (neutral.0): #ffffff (Luminanz: 1.0)
- Neutral.50: #fafafa (Luminanz: 0.98)
- Kontrast: 1.02:1 (sehr subtil, aber wahrnehmbar)

Dieser Kontrast reicht aus, um Zeilen zu unterscheiden, ohne visuelle Unruhe zu erzeugen.

## Row Hover

Zusätzlich zu Zebra-Striping gibt es einen Hover-State:

```
tbody tr:hover {
  background: var(--table-row-hover); /* hydrophon.blau.50 */
  transition: background 150ms ease;
}
```

### Verhalten:

- Überschreibt Zebra-Striping bei Hover
- Hydrophon-Blau für Marken-Konsistenz
- Smooth Transition (150ms)

## Responsive-Verhalten

### Option 1: Horizontaler Scroll (empfohlen)

Tabelle bleibt semantisch, scrollt horizontal auf kleinen Screens:

```
<div class="table-container">
  <table class="data-table">
    <!-- ... -->
  </table>
</div>
```

```
.table-container {
  overflow-x: auto;
  -webkit-overflow-scrolling: touch; /* iOS smooth scrolling */
}

.data-table {
  min-width: var(--table-responsive-min-width); /* 600px */
}
```

**Vorteile:**

- Behält semantische Struktur für Screen Reader
- Einfache Implementierung
- Funktioniert für komplexe Tabellen

**Scroll-Indikatoren:**

Zeige visuellen Hinweis, dass Tabelle scrollbar ist:

```
.table-container {  
  position: relative;  
  overflow-x: auto;  
}  
  
/* Shadow am rechten Rand wenn scrollbar */  
.table-container::after {  
  content: '';  
  position: absolute;  
  top: 0;  
  right: 0;  
  bottom: 0;  
  width: 40px;  
  background: linear-gradient(to left, rgba(0,0,0,0.1), transparent);  
  pointer-events: none;  
  opacity: 1;  
}  
  
/* Shadow ausblenden wenn komplett gescrollt */  
.table-container::-webkit-scrollbar-thumb {  
  /* Custom scrollbar styling */  
}
```

**Option 2: Stacked Cards (für einfache Tabellen)**

Tabelle wird zu Cards auf Mobile umgebaut:

```

@media (max-width: 640px) {
  .data-table thead {
    position: absolute;
    width: 1px;
    height: 1px;
    padding: 0;
    margin: -1px;
    overflow: hidden;
    clip: rect(0, 0, 0, 0);
    white-space: nowrap;
    border-width: 0;
  }

  .data-table tbody tr {
    display: block;
    margin-bottom: var(--spacing-4);
    border: 1px solid var(--neutral-200);
    border-radius: var(--border-radius-md);
  }

  .data-table td {
    display: block;
    text-align: right;
    padding: var(--spacing-3) var(--spacing-4);
    border-bottom: 1px solid var(--neutral-100);
  }

  .data-table td::before {
    content: attr(data-label);
    float: left;
    font-weight: var(--font-weight-semibold);
    color: var(--color-text-primary);
  }

  .data-table td:last-child {
    border-bottom: none;
  }
}

```

HTML mit data-label:

```

<tr>
  <th scope="row">Ablaufrinne 80cm</th>
  <td data-label="Länge">80cm</td>
  <td data-label="Breite">10cm</td>
  <td data-label="Material">Edelstahl</td>
  <td data-label="Preis">189,00 €</td>
</tr>

```

Nachteile:

- Komplexer CSS
- Funktioniert nur für einfache Tabellen
- Nicht geeignet für viele Spalten

**Empfehlung:** Nutze Option 1 (Horizontal Scroll) für Hydrophon-Produkttabellen, da diese oft viele technische Spezifikationen enthalten.

## Caption

<caption> beschreibt Tabelleninhalt für Screen Reader:

```
<table>
  <caption>Technische Spezifikationen Hydrophon Ablaufrinnen</caption>
  <thead>...</thead>
  <tbody>...</tbody>
</table>
```

Styling:

```
caption {
  font-size: var(--table-caption-font-size); /* 18px */
  font-weight: var(--table-caption-font-weight); /* 600 */
  color: var(--table-caption-color);
  text-align: var(--table-caption-text-align); /* left */
  padding: var(--table-caption-padding);
  caption-side: top;
}
```

### Warum wichtig?

- Screen Reader lesen Caption als erstes
- Gibt Kontext über Tabelleninhalt
- WCAG 1.3.1 - Info and Relationships

## Tokens

Data Tables nutzen Tokens aus tokens/tables.json :

Token	Wert	Verwendung
table.fontSize	14px (fontSize.sm)	Basis-Schriftgröße
table.background	white (neutral.0)	Tabellen-Hintergrund
table.borderColor	neutral.200	Borders
table.borderRadius	6px (borderRadius.md)	Container-Ecken
table.cell.paddingY	12px (spacing.3)	Vertikales Cell-Padding
table.cell.paddingX	16px (spacing.4)	Horizontales Cell-Padding
table.header.background	neutral.100	Header-Hintergrund
table.header.color	neutral.900	Header-Text
table.header.fontWeight	600 (semibold)	Header-Gewicht
table.header.borderBottom.width	2px	Header-Border unten
table.header.borderBottom.color	neutral.300	Header-Border-Farbe
table.row.default	transparent	Odd-Row-Hintergrund
table.row.striped	neutral.50	Even-Row-Hintergrund
table.row.hover	hydrophon.blau.50	Hover-Hintergrund
table.row.borderBottom.width	1px	Row-Border unten
table.row.borderBottom.color	neutral.200	Row-Border-Farbe
table.caption.fontSize	18px (fontSize.lg)	Caption-Größe
table.caption.fontWeight	600 (semibold)	Caption-Gewicht
table.caption.padding	16px (spacing.4)	Caption-Padding
table.responsive minWidth	600px	Min-Width vor Scroll

## Code-Beispiele

### Basis-Tabelle

```





```

```
</tbody>  
</table>
```

## CSS-Implementierung

```
/* Table Container für horizontales Scroll */
.table-container {
  overflow-x: auto;
  border-radius: var(--table-border-radius);
  border: 1px solid var(--table-border-color);
  -webkit-overflow-scrolling: touch; /* iOS smooth scrolling */
}

/* Table */
.data-table {
  width: 100%;
  min-width: var(--table-responsive-min-width);
  border-collapse: collapse;
  font-size: var(--table-font-size);
  background: var(--table-background);
}

/* Caption */
.data-table caption {
  font-size: var(--table-caption-font-size);
  font-weight: var(--table-caption-font-weight);
  color: var(--table-caption-color);
  text-align: var(--table-caption-text-align);
  padding: var(--table-caption-padding);
  caption-side: top;
}

/* Header */
.data-table thead {
  background: var(--table-header-background);
}

.data-table th {
  padding: var(--table-cell-padding-y) var(--table-cell-padding-x);
  text-align: left;
  font-weight: var(--table-header-font-weight);
  color: var(--table-header-color);
}

.data-table thead th {
  border-bottom: var(--table-header-border-bottom-width) solid var(--table-header-border-bottom-color);
}

/* Body Cells */
.data-table td {
  padding: var(--table-cell-padding-y) var(--table-cell-padding-x);
  color: var(--color-text-primary);
}

.data-table tbody th {
  font-weight: var(--font-weight-semibold);
  text-align: left;
}

/* Row Borders */
```

```
.data-table tbody tr {
  border-bottom: var(--table-row-border-bottom-width) solid var(--table-row-border-bottom-color);
}

.data-table tbody tr:last-child {
  border-bottom: none;
}

/* Zebra-Striping */
.data-table tbody tr:nth-child(odd) {
  background: var(--table-row-default);
}

.data-table tbody tr:nth-child(even) {
  background: var(--table-row-stripe);
}

/* Hover State */
.data-table tbody tr:hover {
  background: var(--table-row-hover);
  transition: background 150ms ease;
}

/* Numeric Column Alignment */
.data-table td[data-type="number"] {
  text-align: right;
  font-variant-numeric: tabular-nums; /* Monospace numbers */
}
```

## Responsive mit Scroll-Indikatoren

```
<div class="table-container" data-scrollable>
  <table class="data-table">
    <!-- ... -->
  </table>
</div>
```

```

.table-container {
  position: relative;
  overflow-x: auto;
}

/* Shadow am rechten Rand wenn nicht komplett gescrollt */
.table-container::after {
  content: '';
  position: absolute;
  top: 0;
  right: 0;
  bottom: 0;
  width: 30px;
  background: linear-gradient(to left, rgba(0,0,0,0.08), transparent);
  pointer-events: none;
  opacity: 0;
  transition: opacity 200ms;
}

.table-container[data-scrollable]::after {
  opacity: 1;
}

/* Shadow ausblenden am linken Rand */
.table-container::before {
  content: '';
  position: absolute;
  top: 0;
  left: 0;
  bottom: 0;
  width: 30px;
  background: linear-gradient(to right, rgba(0,0,0,0.08), transparent);
  pointer-events: none;
  opacity: 0;
  transition: opacity 200ms;
  z-index: 1;
}

```

#### JavaScript für Scroll-Detection:

```

const tableContainer = document.querySelector('.table-container');

tableContainer.addEventListener('scroll', () => {
  const isScrolledToEnd = tableContainer.scrollWidth - tableContainer.clientWidth <= tableContainer.clientWidth;
  const isScrolledToStart = tableContainer.scrollLeft <= 1;

  if (isScrolledToEnd) {
    tableContainer.removeAttribute('data-scrollable');
  } else {
    tableContainer.setAttribute('data-scrollable', '');
  }
});

```

## Mit Sortierung (optional)

```
<thead>
  <tr>
    <th scope="col">
      <button class="table-sort" aria-label="Sortiere nach Produkt">
        Produkt
        <svg class="table-sort__icon"><!-- Lucide ChevronDown --></svg>
      </button>
    </th>
    <th scope="col">
      <button class="table-sort" aria-label="Sortiere nach Länge">
        Länge
        <svg class="table-sort__icon"><!-- Lucide ChevronDown --></svg>
      </button>
    </th>
  </tr>
</thead>
```

```
.table-sort {
  display: flex;
  align-items: center;
  gap: var(--spacing-2);
  width: 100%;
  padding: 0;
  background: transparent;
  border: none;
  font: inherit;
  font-weight: var(--font-weight-semibold);
  color: inherit;
  cursor: pointer;
}

.table-sort:hover {
  color: var(--hydrophon-blau-600);
}

.table-sort__icon {
  width: 16px;
  height: 16px;
  transition: transform 200ms;
}

.table-sort[aria-sort="ascending"] .table-sort__icon {
  transform: rotate(180deg);
}

.table-sort:focus-visible {
  outline: 2px solid var(--hydrophon-blau-500);
  outline-offset: 2px;
}
```

## Barrierefreiheit

### WCAG 1.3.1 - Info and Relationships

scope-Attribut auf allen Header-Zellen:

```
<th scope="col">Spalte</th> <!-- Spalten-Header -->
<th scope="row">Zeile</th> <!-- Zeilen-Header -->
```

Screen Reader ordnen dadurch Daten dem korrekten Header zu:

- Nutzer navigiert zu Zelle "80cm"
- Screen Reader liest: "Ablaufrinne 80cm, Länge: 80cm"

### caption für Tabellenbeschreibung

```
<caption>Technische Spezifikationen Hydrophon Ablaufrinnen</caption>
```

Screen Reader lesen Caption als erstes, bevor sie in Tabelle navigieren.

### Keine merged cells ohne headers-Attribut

Vermeiden:

```
<!-- FALSCH: Screen Reader können Beziehung nicht erkennen -->
<td colspan="3">Gesamtpreis: 999,00 €</td>
```

Wenn merged cells notwendig, nutze headers -Attribut:

```
<th id="gesamt">Gesamtpreis</th>
<td headers="gesamt" colspan="3">999,00 €</td>
```

## Keyboard-Navigation

- **Tab:** Navigiert durch interaktive Elemente (Links, Buttons)
- **Arrow Keys:** Screen Reader Table Navigation (Zeile/Spalte)
- **Ctrl + Alt + Arrow:** NVDA/JAWS Table Navigation

Keine Custom Keyboard-Navigation erforderlich, da native `<table>` verwendet wird.

### Sortierung mit aria-sort

Falls Tabelle sortierbar:

```
<th scope="col" aria-sort="ascending">
  <button>Länge</button>
</th>
```

#### aria-sort-Werte:

- ascending – Aufsteigend sortiert
- descending – Absteigend sortiert
- none – Nicht sortiert (default)

## Best Practices

### Dos

- Immer scope-Attribut nutzen – scope="col" und scope="row"
- caption für Tabellenbeschreibung – Screen Reader Context
- Subtile Zebra-Stripes – neutral.50 (nicht stärker)
- Hover-State für Zeilen – Verbessert Interaktivität
- Horizontales Scroll auf Mobile – Behält Semantik
- Numeric Alignment rechts – text-align: right für Zahlen
- tabular-nums für Zahlen – font-variant-numeric: tabular-nums

### Don'ts

- Keine div-Tabellen – Immer semantisches <table> nutzen
- Keine fehlenden Header – Jede Spalte braucht <th scope="col">
- Keine starken Zebra-Kontraste – Max neutral.100, besser neutral.50
- Keine merged cells ohne headers – Screen Reader verlieren Kontext
- Keine fixed Spaltenbreiten – Lässt keine Flexibilität auf Mobile
- Keine Tabellen für Layout – Nur für tabellarische Daten

## Sortierung (optional)

Falls Tabellen sortierbar sein sollen, kann JavaScript-Sortierung implementiert werden:

HTML:

```
<thead>
  <tr>
    <th scope="col" data-sortable data-type="text">Produkt</th>
    <th scope="col" data-sortable data-type="number">Länge</th>
    <th scope="col" data-sortable data-type="number">Preis</th>
  </tr>
</thead>
```

JavaScript (Beispiel):

```

document.querySelectorAll('[data-sortable]').forEach(th => {
  th.addEventListener('click', () => {
    const table = th.closest('table');
    const tbody = table.querySelector('tbody');
    const rows = Array.from(tbody.querySelectorAll('tr'));
    const columnIndex = Array.from(th.parentNode.children).indexOf(th);
    const type = th.dataset.type || 'text';
    const currentSort = th.getAttribute('aria-sort') || 'none';
    const newSort = currentSort === 'ascending' ? 'descending' : 'ascending';

    // Reset all other columns
    th.parentNode.querySelectorAll('[aria-sort]').forEach(t => {
      if (t !== th) t.removeAttribute('aria-sort');
    });

    th.setAttribute('aria-sort', newSort);

    // Sort rows
    rows.sort((a, b) => {
      const aVal = a.children[columnIndex].textContent.trim();
      const bVal = b.children[columnIndex].textContent.trim();

      if (type === 'number') {
        return newSort === 'ascending'
          ? parseFloat(aVal) - parseFloat(bVal)
          : parseFloat(bVal) - parseFloat(aVal);
      } else {
        return newSort === 'ascending'
          ? aVal.localeCompare(bVal)
          : bVal.localeCompare(aVal);
      }
    });

    // Re-append sorted rows
    rows.forEach(row => tbody.appendChild(row));
  });
});

```

## Verwandte Komponenten

- Product Card – Alternative Darstellung für Produkt-Übersichten
- Content Card – Card-basierte Layouts

## Ressourcen

- MDN: `<table>` Element
- MDN: `scope` Attribut
- WCAG 1.3.1 - Info and Relationships
- WebAIM: Creating Accessible Tables
- CSS Tricks: Responsive Tables

# Feedback & Systemrueckmeldungen

---

Vollstaendiges Feedback-System fuer Benutzerinteraktionen mit Fokus auf Klarheit und Barrierefreiheit.

## Uebersicht

Dieses Kapitel dokumentiert alle Feedback-Komponenten des Hydrophon Design Systems. Feedback-Komponenten kommunizieren Systemzustaende, bestaetigen Nutzeraktionen und leiten durch komplexe Interaktionen.

**Kernprinzip:** Feedback sollte **nicht-aufdringlich** sein – es unterstuetzt Nutzer, statt sie zu unterbrechen.

## Design-Philosophie

### 1. Hierarchie der Aufmerksamkeit

Nicht jedes Feedback ist gleich wichtig. Unsere Komponenten folgen einer klaren Hierarchie:

Komponente	Aufmerksamkeit	Verwendung
Modal	Hoch (blockierend)	Destruktive Aktionen, wichtige Entscheidungen
Toast	Mittel (sichtbar, nicht blockierend)	Aktionsbestaetigung, Fehler mit Retry
Tooltip	Niedrig (bei Bedarf)	Kontextuelle Labels, Erklaerungen
Loading	Kontextabhaengig	Fortschritt, Ladezustaende

### 2. Non-Intrusive Feedback

- Feedback stoert den Workflow nicht unnoetig
- Toasts statt Modals wo moeglich
- Inline-Feedback (Formular-Validierung) vor globalen Meldungen
- Optimistic UI fuer schnelle gefuehlte Performance

### 3. Accessibility First

Alle Feedback-Komponenten erfuellen **WCAG 2.1 AA**:

- ARIA-Attribute fuer Screenreader
- Tastatur-Navigation
- prefers-reduced-motion Unterstuetzung
- Farbunabhaengige visuelle Indikatoren

## Komponenten

### Modal-Dialoge

**Zweck:** Fokussierte Interaktionen, die volle Aufmerksamkeit erfordern

**Dokumentation:** [Modal-Dialoge](#)

**Verwendung:**

- **Bestaetigungsdialoge:** "Wirklich loeschen?" mit Ja/Abbrechen

- **Formulare:** Kontaktformular, Benutzer anlegen
- **Informationsdialoge:** Datenschutzhinweis, Willkommensnachricht

**Eigenschaften:**

- Drei Groessen (sm/md/lg)
- ESC-Key + Close-Button (X)
- KEIN Backdrop-Click (verhindert versehentliches Schliessen)
- Focus-Trap Pattern
- Overlay mit halbtransparentem Schwarz
- Zentrale Positionierung

**Wann NICHT verwenden:**

- Einfache Bestaetigung (verwende Toast)
  - Nicht-kritische Info (verwende Tooltip)
  - Haeufige Aktionen (zu stoerend)
- 

**Tooltips****Zweck:** Kurze kontextuelle Labels fuer Icons und Buttons**Dokumentation:** [Tooltips](#)**Verwendung:**

- Icon-Button-Labels ("Loeschen", "Bearbeiten", "Exportieren")
- Kurze Erklaerungen (1-5 Woerter)
- Supplemental Info (nicht kritisch)

**Eigenschaften:**

- Hover-Trigger (~300ms Delay)
- Smart Positioning (top bevorzugt, flipped bei Viewport-Edge)
- Pfeil-Indikator zeigt auf Trigger-Element
- Dunkler Hintergrund (neutral.900), weisser Text
- Nicht fuer Touch-Geraete (keine kritischen Infos)

**Wann NICHT verwenden:**

- Kritische Informationen (immer sichtbar machen)
  - Lange Erklaerungen (verwende Helptext oder Modal)
  - Mobile-First Content (Tooltips auf Touch schwierig)
- 

**Toast-Benachrichtigungen****Zweck:** Nicht-blockierende Bestaetigung von Aktionen**Dokumentation:** [Toast-Benachrichtigungen](#)**Verwendung:**

- Erfolgsbestaetigung ("Aenderungen gespeichert")
- Fehler mit Retry-Option ("Upload fehlgeschlagen – Erneut versuchen")
- Info-Meldungen ("Neue Nachricht erhalten")
- Warnungen ("Verbindung instabil")

**Eigenschaften:**

- Position: Top-Right
- Auto-Dismiss nach Schweregrad:
  - Success: 3s
  - Info: 4s
  - Warning: 5s
  - Error: Manuell (bleibt bis geschlossen)
- 1-2 Action-Buttons (Undo, Retry, Details)
- Stacken vertikal (max 4 gleichzeitig)

#### Wann NICHT verwenden:

- Formular-Validierungsfehler (inline statt Toast)
  - Kritische Entscheidungen (Modal statt Toast)
  - Sehr haeufige Meldungen (stoeren Workflow)
- 

## Ladezustaende

**Zweck:** Nutzer waehrend asynchroner Operationen informiert halten

**Dokumentation:** [loading.md](#)

**Drei Arten:**

1. **Spinner** – Kurze unbestimmte Aktionen (<3s)
2. **Progress Bar** – Laengere Aktionen mit bekannter/unbekannter Dauer
3. **Skeleton Screens** – Content-Laden (Cards, Tabellen, Listen)

**Verwendung:**

Kontext	Dauer	Indikator	Beispiel
Button-Aktion	<1s	Spinner (delayed)	Formular absenden
API-Call	1-3s	Spinner (immediate)	Daten laden
Upload	>3s, bekannt	Progress Bar (determinate)	Datei hochladen
Verarbeitung	>3s, unbekannt	Progress Bar (indeterminate)	PDF generieren
Content-Laden	Beliebig	Skeleton Screen	Produktkarten

**Eigenschaften:**

- Spinner: 4 Groessen (sm/md/lg/xl), 200ms Delay gegen Flash
- Progress Bar: Native `<progress>` Element, determinate/indeterminate
- Skeleton: react-loading-skeleton, matcht finales Layout exakt
- Optimistic UI fuer Like/Save-Aktionen
- prefers-reduced-motion Unterstuetzung

**Wann welchen Indikator:**

- **Spinner:** "Ich arbeite daran" (unbestimmt)
- **Progress Bar:** "Ich bin zu X% fertig" (bestimmt) oder "Ich arbeite, weiss aber nicht wie lange" (unbestimmt)
- **Skeleton:** "So wird der Content aussehen" (Struktur-Vorschau)

## Wann welche Komponente

Entscheidungsmatrix fuer Feedback-Komponenten:

Use Case	Komponente	Begründung
Destruktive Aktion bestaetigen (Loeschen)	Modal	Volle Aufmerksamkeit noetig, verhindert Fehler
Aenderungen gespeichert	Toast (Success)	Bestaetigung ohne Unterbrechung
Upload fehlgeschlagen	Toast (Error) + Retry	Nicht-blockierend, bietet Loesung
Icon-Button erklaeren	Tooltip	Kurzes Label bei Bedarf
Formular mit Fehler	Inline Error + Icon	Nah am Problem, nicht global
Seite laedt	Skeleton Screen	Zeigt Struktur, vermeidet Layout Shift
Button-Aktion verarbeiten	Spinner (delayed)	Zeigt Aktivitaet, vermeidet Flash
Datei hochladen	Progress Bar (determinate)	Zeigt Fortschritt, beruhigt User
Komplexes Formular	Modal	Fokussierte Eingabe ohne Ablenkung
Hilfetext zu Eingabefeld	Helper Text (inline)	Immer sichtbar, nicht versteckt
"Erfolgreich angelegt"	Toast (Success) + Redirect	Kurze Bestaetigung, dann weiter
Verbindungsfehler	Toast (Error) + Retry	Persistent bis Loesung

## Entscheidungsbaum

```

Braucht Nutzer sofortige Entscheidung zu treffen?
└ Ja → Ist Aktion destruktiv/kritisch?
  | └ Ja → MODAL (Bestaetigungsdialog)
  | └ Nein → TOAST mit Action-Buttons
  └ Nein → Ist es eine Bestaetigung?
    | └ Ja → TOAST (Auto-Dissmiss)
    | └ Nein → Ist es kontextuelle Info?
      | └ Ja → TOOLTIP (bei Hover)
      | └ Nein → Ist System am Laden?
        | └ Ja → LOADING (Spinner/Progress/Skeleton)
        | └ Sonst → Inline-Feedback (Text/Icon)
  
```

## Gemeinsame Prinzipien

Alle Feedback-Komponenten teilen diese Design-Prinzipien:

### 1. Klarheit

Feedback muss eindeutig sein:

- Erfolg → Gruen + Checkmark + "Gespeichert"
- Fehler → Rot + X + "Fehler: [was schiefging]" + "Wie beheben"
- Warning → Gelb + Alert-Icon + "Achtung: [was beachten]"
- Info → Blau + Info-Icon + "Hinweis: [zusaetzliche Info]"

**Nicht nur Farbe:** Immer Icon + Text kombinieren (WCAG 1.4.1).

## 2. Timing

Auto-Dissmiss-Regeln:

Typ	Dauer	Begründung
Success	3s	Kurz – Nutzer weiss, was passiert ist
Info	4s	Etwas laenger – evtl. neue Info zu lesen
Warning	5s	Laenger – Nutzer soll aufmerksam werden
Error	Manuell	Bleibt bis geschlossen – Nutzer muss reagieren

**Verzoegerung (Delay):**

- Spinner: 200ms Delay (vermeidet Flash bei schnellen Ops)
- Tooltip: 300ms Delay (vermeidet versehentliches Triggern)

## 3. Konsistente Positionierung

**Modals:** Zentriert (vertikal & horizontal) **Toasts:** Top-Right (nicht blockierend, konsistent mit Desktop-Apps) **Tooltips:**

Smart Positioning (Preferenz: oben, flipped bei Bedarf) **Loading:**

- Inline-Spinner: Im Button/Container selbst
- Page-Spinner: Zentriert ueber Content
- Skeleton: Ersetzt Content-Bereich 1:1

## 4. Animation

**Einheitliche Animations-Dauer:**

- Modal: 200ms (ease-out)
- Toast: Slide + Fade (Sonner default)
- Tooltip: Fade 150ms (schnell, nicht störend)
- Spinner: 750ms Rotation (linear)
- Skeleton Shimmer: 1.5s (ease-in-out)

**prefers-reduced-motion:**

Alle Animationen haben Fallback fuer Nutzer mit Bewegungsempfindlichkeit:

```
@media (prefers-reduced-motion: reduce) {
  .modal {
    animation: none;
    opacity: 1; /* Sofort sichtbar */
  }

  .spinner {
    animation: none;
    opacity: 0.5; /* Statisch, halbtransparent */
  }
}
```

## 5. Barrierefreiheit

### ARIA-Patterns:

- Modals: `role="dialog"`, `aria-modal="true"`, `aria-labelledby`, Focus-Trap
- Toasts: `role="status"` (Success/Info) oder `role="alert"` (Error/Warning), `aria-live="polite"`
- Tooltips: `role="tooltip"`, `aria-describedby` Verknuepfung
- Loading: `role="status"`, `aria-live="polite"`, `aria-busy="true"`

### Tastatur-Navigation:

- Modal: Tab-Cycling innerhalb, ESC schliesst
- Toast: Fokussierbar, Space/Enter fuer Action-Buttons
- Tooltip: Fokussierbar via Keyboard (nicht nur Hover)
- Spinner: Nicht fokussierbar (Container hat focus)

### Screenreader-Ansagen:

- Modal-oeffnen: Focus auf ersten Heading/Close-Button
- Toast: Live-Region kuendigt Inhalt an
- Loading-Abschluss: "X Produkte geladen" Ansage
- Error: Alert-Region zieht sofort Aufmerksamkeit

## Integration

Alle Feedback-Komponenten verwenden Token aus `design-system/tokens/feedback.json`:

```
import tokens from '../design-system/build/json/tokens.json';

// Modal-Sizing
const modalWidth = tokens.modal.size.md.width.$value; // "600px"

// Toast-Duration
const toastDuration = tokens.toast.duration.success.$value; // "3000ms"

// Spinner-Size
const spinnerSize = tokens.spinner.size.md.$value; // "24px"

// Skeleton-Colors
const skeletonBase = tokens.skeleton['base-color'].$value; // "#e5e5e5"
```

### CSS Custom Properties:

Alle Tokens werden von Style Dictionary zu CSS-Variablen kompiliert:

```

.modal {
  width: var(--modal-size-md-width);
  padding: var(--modal-content-padding-x) var(--modal-content-padding-y);
  border-radius: var(--modal-content-border-radius);
  box-shadow: var(--modal-content-shadow);
}

.toast-success {
  border-left: 4px solid var(--toast-border-success);
  padding: var(--toast-container-padding);
}

.spinner-md {
  width: var(--spinner-size-md);
  height: var(--spinner-size-md);
  border-width: var(--spinner-stroke-width);
  animation-duration: var(--spinner-animation-duration);
}

```

## Best Practices

### Do's

- **Modal fuer kritische Entscheidungen** (Loeschen, Verwerfen, Verlassen)
- **Toast fuer Aktionsbestaetigung** (Gespeichert, Geloescht mit Undo)
- **Tooltip fuer Icon-Labels** (kurz, supplemental)
- **Skeleton fuer Content-Laden** (zeigt Struktur, vermeidet Shift)
- **Spinner mit 200ms Delay** (vermeidet Flash)
- **Progress Bar fuer lange Ops** (Upload, Download, Processing)
- **Optimistic UI fuer Like/Save** (sofortiges Feedback)
- **Error mit Retry-Button** (gibt Nutzer Kontrolle)
- **prefers-reduced-motion** (immer implementieren)

### Don'ts

- **Kein Modal fuer Simple Bestaetigung** (Toast reicht)
- **Kein Toast fuer Formular-Fehler** (Inline Error am Field)
- **Kein Tooltip fuer kritische Info** (immer sichtbar machen)
- **Kein Backdrop-Click** (versehentliches Schliessen)
- **Keine Auto-Dismis Errors** (Nutzer muss lesen koennen)
- **Kein Spinner ohne Delay** (Flash-Problem)
- **Keine Hardcoded Skeleton-Groessen** (Layout Shift)
- **Kein Optimistic UI fuer Destruktive Aktionen** (zu riskant)
- **Keine Animationen ohne prefers-reduced-motion Fallback**

## Verwandte Phasen

### Phase 1: Foundation & Brand Identity

- Farben (Success/Error/Warning/Info) → Feedback-Farben
- Typografie → Modal/Toast Schriftgroessen
- Spacing → Padding/Margins in Feedback-Komponenten

### Phase 2: Icons & Basic Interactions

- Icons → Toast Icons (Check, X, Alert, Info)
- Button-System → Modal Action-Buttons, Toast Action-Buttons

### Phase 3: Forms & Data Input

- Form-Validierung → Toast Error-Pattern
- Input States → Konsistente Error-Darstellung

### Phase 4: Navigation & Content Structure

- Cards → Skeleton Patterns (ProductCardSkeleton)
- Tables → Skeleton Patterns (TableSkeleton)
- Focus-Trap → Modal Focus-Management

## Weiterfuehrende Ressourcen

### WCAG Guidelines

- [WCAG 2.1.2 - No Keyboard Trap](#)
- [WCAG 1.4.1 - Use of Color](#)
- [WCAG 1.4.13 - Content on Hover or Focus](#)
- [WCAG 2.2.4 - Interruptions](#)

### ARIA Patterns

- [ARIA: Dialog \(Modal\) Pattern](#)
- [ARIA: Tooltip Pattern](#)
- [ARIA: Alert Pattern](#)
- [ARIA: Status Role](#)

### Libraries

- [Radix UI Primitives](#) – Headless UI fuer Modal/Tooltip
- [Sonner](#) – Modern Toast Library
- [react-loading-skeleton](#) – Auto-sizing Skeleton Screens

### Articles

- [Nielsen Norman: Progress Indicators](#)
- [Smashing Magazine: Skeleton Screens](#)

- [Web.dev: Optimistic UI](#)
- 

**Phase 5 abgeschlossen:** Feedback & System Responses vollstaendig dokumentiert mit 4 Komponenten (Modal, Tooltip, Toast, Loading) und 150+ Tokens.

Letzte Aktualisierung: 2026-01-29 Version: 1.0

---

# Modal-Dialoge

---

Modale Dialoge unterbrechen den Hauptfluss und fordern den Fokus des Nutzers für wichtige Aktionen, Bestätigungen oder Formulareingaben.

**Anforderungen:** FEEDBACK-01 (Modal-Varianten), FEEDBACK-02 (Accessibility), FEEDBACK-03 (Focus-Management), FEEDBACK-04 (Keyboard-Support)

---

## Übersicht

Ein Modal-Dialog ist ein Overlay-Fenster, das über dem Hauptinhalt erscheint und die Aufmerksamkeit des Nutzers erfordert. Der Hintergrund wird abgedunkelt und nicht-interaktiv, bis der Dialog geschlossen wird.

### Wann Modals verwenden:

- **Bestätigungsdialoge** — Kritische oder destruktive Aktionen absichern (z.B. "Artikel wirklich löschen?")
- **Formular-Dialoge** — Kurze Formulare ohne neue Seite (z.B. "Neuen Kontakt anlegen")
- **Informationsdialoge** — Wichtige Informationen präsentieren, die Aufmerksamkeit erfordern

### Wann KEINE Modals verwenden:

- **Komplexe Workflows** — Mehrschrittige Prozesse gehören auf dedizierte Seiten
- **Nebenläufige Interaktion** — Wenn Nutzer parallel arbeiten müssen (dann Slide-Out-Panel)
- **Nicht-kritische Infos** — Standard-Inhalte gehören inline oder auf separate Seiten
- **Häufige Aktionen** — Modal-Overload frustriert Nutzer

### B2B-Kontext:

B2B-Nutzer arbeiten oft mit komplexen Datensätzen. Modals eignen sich für schnelle Bestätigungen und kurze Formulare, aber nicht für umfangreiche Dateneingabe. Im Zweifelsfall: dedizierte Seite statt Modal.

---

## Varianten

Das Design-System definiert drei Modal-Größen für unterschiedliche Anwendungsfälle:

### Small (400px)

**Verwendung:** Bestätigungsdialoge, einfache Ja/Nein-Entscheidungen

#### Typischer Inhalt:

- Titel (1 Zeile)
- Beschreibung (2-3 Zeilen)
- 2 Buttons (Abbrechen + Aktion)

#### Beispiel:

[X] Artikel löschen?

Dieser Vorgang kann nicht rückgängig gemacht werden.

[Abbrechen] [Löschen]

Token: modal.container.max-width.sm → 400px

### Medium (600px)

**Verwendung:** Formulare mit 3-8 Feldern, mittlere Inhaltsmengen

**Typischer Inhalt:**

- Titel
- Formularfelder (Text-Inputs, Selects, Checkboxen)
- Hilfetext
- 2-3 Buttons (Abbrechen + Speichern + optional Mehr)

**Beispiel:**

[X] Neuen Kontakt anlegen

Name: [\_\_\_\_\_]

E-Mail: [\_\_\_\_\_]

Telefon: [\_\_\_\_\_]

Rolle: [▼ Bitte wählen]

[Abbrechen] [Speichern]

Token: modal.container.max-width.md → 600px

### Large (900px)

**Verwendung:** Komplexe Inhalte, Vorschau-Dialoge, mehrspaltige Formulare

**Typischer Inhalt:**

- Titel
- Mehrspaltige Layouts
- Tabellen oder Listen
- Bild-Galerien

- Komplexe Formulare (>8 Felder)

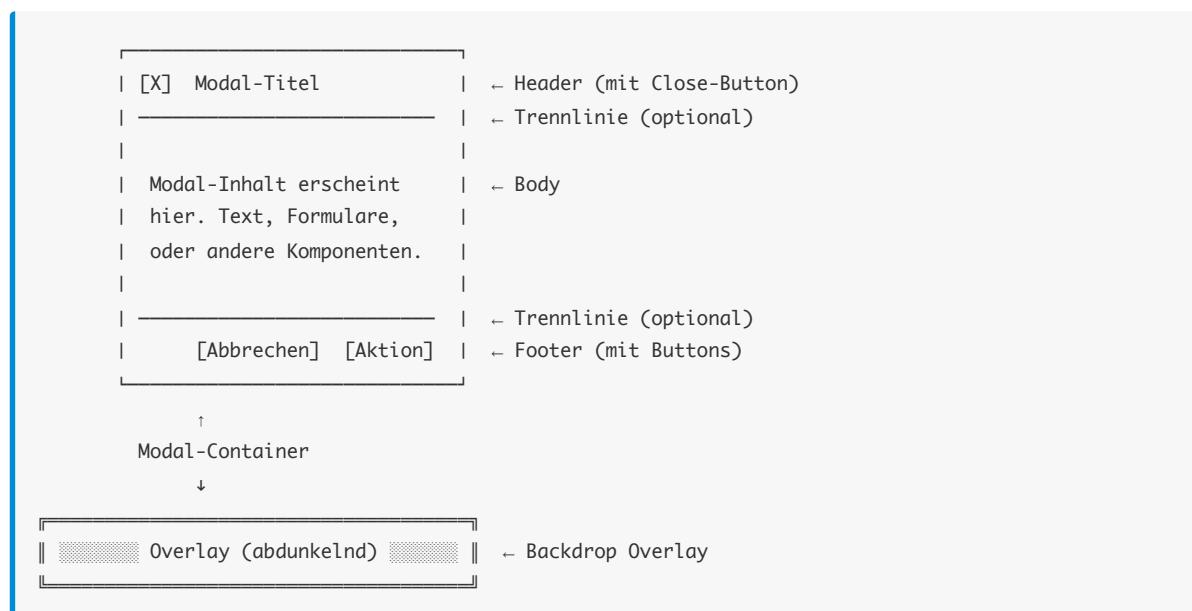
Beispiel:

The screenshot shows a modal window with a light gray background and a white content area. At the top left is a close button [X]. Below it is a section titled "Produktdetails". Inside this section, there are several input fields and labels. One field has the placeholder "Name: Hydrophon Serie A". Another field has "Artikelnr.: HYD-A-001". There is also a field with "[Bild]" and another with "Preis: 249,00 €". Other fields include "Verfügbar: Ja" and "Kategorie: Armaturen". Below this section is a "Beschreibung:" label followed by a long text block starting with "Lorem ipsum dolor sit amet, consectetur adipiscing...". Underneath this is a "Spezifikationen:" label with a list of bullet points: "Material: Edelstahl", "Durchfluss: 12 L/min", "Montage: Wandmontage", and "Höhe: 150mm". At the bottom right of the modal are two buttons: "[Schließen]" and "[In Warenkorb]".

Token: modal.container.max-width.lg → 900px

## Anatomie

Ein Modal besteht aus fünf Elementen:



## 1. Overlay (Backdrop)

**Funktion:** Visueller Kontext, signalisiert modalen Zustand

**Tokens:**

- Hintergrund: `modal.overlay.background` (rgba(0,0,0,0.5))

**Verhalten:**

- Deckung: Gesamter Viewport
- Klick: KEINE Schließen-Funktion (verhindert versehentliches Schließen)
- Z-Index: Unter Modal-Container

**Warum kein Backdrop-Click?**

Versehentliches Schließen ist frustrierend, besonders bei Formularen mit ungespeicherten Änderungen. B2B-Nutzer arbeiten oft in hektischen Umgebungen (Baustelle, Lager). Ein Fehlklick auf den Backdrop würde Datenverlust riskieren.

## 2. Modal-Container

**Funktion:** Enthält den gesamten Dialog-Inhalt

**Tokens:**

- Hintergrund: `modal.container.background` (neutral.white)
- Eckenradius: `modal.container.border-radius` (borderRadius.lg / 8px)
- Padding: `modal.container.padding` (spacing.6 / 24px)
- Breite: `modal.container.max-width.{sm|md|lg}` (400px / 600px / 900px)

**Verhalten:**

- Position: Zentriert im Viewport (horizontal & vertikal)
- Max-Höhe: 80-90vh (verhindert Viewport-Overflow)
- Scroll: Vertikal scrollbar bei langem Inhalt
- Shadow: Elevation für Tiefe (shadow.xl)

## 3. Header mit Close-Button

**Funktion:** Titel + Schließen-Möglichkeit

**Struktur:**

```
<div class="modal__header">
  <h2 class="modal__title">Artikel löschen?</h2>
  <button class="modal__close" aria-label="Dialog schließen">
    <svg><!-- X-Icon --></svg>
  </button>
</div>
```

**Tokens:**

- Titel Font-Size: `modal.title.fontSize` (typography.heading.h3.fontSize / 30px)
- Titel Font-Weight: `modal.title.fontWeight` (typography.heading.h3.fontWeight / semibold)
- Titel Farbe: `modal.title.color` (neutral.900)
- Close-Button Größe: `modal.close-button.size` (32px)
- Close-Button Icon: `modal.close-button.icon-size` (20px)

**Accessibility:**

- Close-Button IMMER sichtbar (nicht nur visuell, auch für Screenreader)
- aria-label auf Close-Button: "Dialog schließen" (beschreibt Aktion)

## 4. Body

**Funktion:** Hauptinhalt des Modals

**Inhalt:** Beliebig — Text, Formulare, Bilder, Tabellen

**Tokens:**

- Padding unten: modal.description.margin-bottom (spacing.6 / 24px)
- Textfarbe: modal.description.color (neutral.600)
- Font-Size: modal.description.font-size (typography.body.md.fontSize / 16px)

**Scroll-Verhalten:**

```
.modal__body {
  max-height: calc(90vh - 200px); /* Abzüglich Header + Footer */
  overflow-y: auto;
}
```

Wenn Body-Inhalt zu lang wird, scrollt nur der Body — Header und Footer bleiben fixiert.

## 5. Footer

**Funktion:** Enthält Aktions-Buttons

**Layout:** Rechtsbündig, primäre Aktion rechts

**Tokens:**

- Padding oben: modal.footer.padding-top (spacing.4 / 16px)
- Button-Gap: modal.footer.gap (spacing.3 / 12px)

**Button-Reihenfolge (Links nach Rechts):**

1. **Tertiär-Button** (optional) — Z.B. "Mehr erfahren"
2. **Sekundär-Button** — "Abbrechen" (nicht-destruktiv)
3. **Primär-Button** — Hauptaktion (z.B. "Speichern", "Löschen")

**Beispiel:**

```
<div class="modal__footer">
  <button class="button button--tertiary">Hilfe</button>
  <button class="button button--secondary">Abbrechen</button>
  <button class="button button--primary">Speichern</button>
</div>
```

## Design-Token

Alle Modal-Tokens aus design-system/tokens/feedback.json :

## Overlay

Token-Name	Wert	CSS-Variable	Beschreibung
modal.overlay.background	rgba(0, 0, 0, 0.5)	--modal-overlay-background	Halbtransparentes Schwarz

## Container

Token-Name	Wert	CSS-Variable	Beschreibung
modal.container.background	{color.neutral.white}	--modal-container-background	Weiß
modal.container.border-radius	{borderRadius.lg}	--modal-container-border-radius	8px
modal.container.padding	{spacing.6}	--modal-container-padding	24px
modal.container.max-width.sm	400px	--modal-container-max-width-sm	Small Modal
modal.container.max-width.md	600px	--modal-container-max-width-md	Medium Modal
modal.container.max-width.lg	900px	--modal-container-max-width-lg	Large Modal

## Close-Button

Token-Name	Wert	CSS-Variable	Beschreibung
modal.close-button.size	32px	--modal-close-button-size	Button-Größe
modal.close-button.color	{color.neutral.600}	--modal-close-button-color	Standard-Farbe
modal.close-button.hover-color	{color.neutral.900}	--modal-close-button-hover-color	Hover-Farbe

## Titel

Token-Name	Wert	CSS-Variable	Beschreibung
modal.title.fontSize	{typography.heading.h3.fontSize}	--modal-title-font-size	30px (H3)
modal.title.fontWeight	{typography.heading.h3.fontWeight}	--modal-title-font-weight	semibold (600)
modal.title.marginBottom	{spacing.4}	--modal-title-margin-bottom	16px

## Beschreibung

Token-Name	Wert	CSS-Variable	Beschreibung
modal.description.fontSize	{typography.body.md.fontSize}	--modal-description-font-size	16px
modal.description.color	{color.neutral.600}	--modal-description-color	Grau
modal.description.marginBottom	{spacing.6}	--modal-description-margin-bottom	24px

## Verhalten

### Öffnen

**Trigger:** Button-Klick, Link-Click, oder programmatisch

**Schritte:**

1. Overlay erscheint mit Fade-In-Animation
2. Modal-Container erscheint mit Fade + Scale-Animation
3. Body-Scroll wird deaktiviert ( `overflow: hidden` auf `<body>` )
4. Fokus bewegt sich auf ersten fokussierbaren Element im Modal (üblicherweise Close-Button)
5. Focus-Trap aktiviert (Tab-Taste zirkuliert innerhalb Modal)

**Animation:**

```
/* Overlay Fade-In */
.modal-overlay {
  animation: fadeIn 200ms ease-out;
}

@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}

/* Modal Scale + Fade */
.modal-container {
  animation: modalEnter 200ms ease-out;
}

@keyframes modalEnter {
  from {
    opacity: 0;
    transform: scale(0.95);
  }
  to {
    opacity: 1;
    transform: scale(1);
  }
}
```

**Timing:** 200ms (aus Token `modal.animation.duration` )

**Reduced Motion:**

```
@media (prefers-reduced-motion: reduce) {
  .modal-overlay,
  .modal-container {
    animation: none;
  }
}
```

## Schließen

Trigger:

- **ESC-Taste** — Keyboard-Escape (WCAG 2.1.2 Pflicht)
- **Close-Button (X)** — Visueller Schließ-Mechanismus
- **Abbrechen-Button** — Im Footer (optional)
- **Erfolgreiche Aktion** — Nach "Speichern", "Löschen" etc.

**NICHT:** Backdrop-Click (verhindert versehentliches Schließen)

Schritte:

1. Modal-Container verschwindet mit Fade + Scale-Out-Animation
2. Overlay verschwindet mit Fade-Out-Animation
3. Fokus kehrt zu auslösendem Element zurück (Button, der Modal geöffnet hat)
4. Focus-Trap deaktiviert
5. Body-Scroll reaktiviert ( `overflow: auto` auf `<body>` )

Animation:

```
/* Modal Scale + Fade Out */
.modal-container--closing {
  animation: modalExit 200ms ease-in;
}

@keyframes modalExit {
  from {
    opacity: 1;
    transform: scale(1);
  }
  to {
    opacity: 0;
    transform: scale(0.95);
  }
}
```

## Focus-Management

Anforderung: WCAG 2.4.3 (Focus Order) + WCAG 2.1.2 (No Keyboard Trap)

Focus-Trap Pattern:

Beim Öffnen wird der Fokus im Modal "gefangen". Tab-Taste zirkuliert durch alle fokussierbaren Elemente innerhalb des Modals.

Implementation (Vanilla JavaScript):

```

function trapFocus(modalElement) {
  const focusableElements = modalElement.querySelectorAll(
    'button:not([disabled]), a[href], input:not([disabled]), select:not([disabled]), textarea:not([disabled])'
  );

  const firstFocusable = focusableElements[0];
  const lastFocusable = focusableElements[focusableElements.length - 1];

  function handleTabKey(event) {
    if (event.key !== 'Tab') return;

    if (event.shiftKey) {
      // Shift + Tab: Vom ersten zum letzten Element springen
      if (document.activeElement === firstFocusable) {
        event.preventDefault();
        lastFocusable.focus();
      }
    } else {
      // Tab: Vom letzten zum ersten Element springen
      if (document.activeElement === lastFocusable) {
        event.preventDefault();
        firstFocusable.focus();
      }
    }
  }

  modalElement.addEventListener('keydown', handleTabKey);

  // Cleanup-Funktion für Event-Listener-Entfernung
  return () => {
    modalElement.removeEventListener('keydown', handleTabKey);
  };
}

```

**Focus-Reihenfolge im Modal:**

1. Close-Button (X) — Erhält initialen Fokus
2. Modal-Body-Inhalt (Links, Inputs, etc.)
3. Footer-Buttons (von links nach rechts)
4. (Tab wraps zurück zu Close-Button)

**Fokus-Rückkehr:**

```

// Trigger-Element speichern
const triggerElement = document.activeElement;

// Modal öffnen
openModal();

// Modal schließen
closeModal();

// Fokus zurück zu Trigger
triggerElement.focus();

```

## ESC-Taste

Anforderung: WCAG 2.1.2 (No Keyboard Trap)

Implementation:

```
document.addEventListener('keydown', (event) => {
  if (event.key === 'Escape' && modalIsOpen) {
    closeModal();
  }
});
```

Ausnahme: Ungespeicherte Änderungen

Wenn Formular-Daten ungespeichert sind, ESC-Taste mit Bestätigungsdialog kombinieren:

```
document.addEventListener('keydown', (event) => {
  if (event.key === 'Escape' && modalIsOpen) {
    if (hasUnsavedChanges) {
      // Bestätigungsdialog zeigen
      if (confirm('Änderungen verwerfen?')) {
        closeModal();
      }
    } else {
      closeModal();
    }
  }
});
```

## Barrierefreiheit (WCAG 2.1 AA)

### ARIA-Attribute

Modal-Dialoge verwenden das WAI-ARIA Dialog Pattern:

Overlay:

```
<div class="modal-overlay" aria-hidden="true"></div>
```

Modal-Container:

```
<div
  class="modal"
  role="dialog"
  aria-modal="true"
  aria-labelledby="modal-title"
  aria-describedby="modal-description"
>
  <h2 id="modal-title">Modal-Titel</h2>
  <p id="modal-description">Modal-Beschreibung</p>
  ...
</div>
```

ARIA-Attribut	Wert/Zweck	Pflicht?
role="dialog"	Identifiziert Element als Dialog	Ja
aria-modal="true"	Kommuniziert modales Verhalten	Ja
aria-labelledby	Verknüpft mit Modal-Titel (ID)	Ja
aria-describedby	Verknüpft mit Beschreibung (ID)	Optional

## Keyboard-Navigation

Anforderungen: WCAG 2.1.1 (Keyboard) + WCAG 2.1.2 (No Keyboard Trap)

Taste	Aktion
Tab	Fokus zum nächsten fokussierbaren Element (innerhalb Modal)
Shift+Tab	Fokus zum vorherigen fokussierbaren Element (innerhalb Modal)
ESC	Modal schließen (Fokus kehrt zu Trigger zurück)
Enter	Primär-Button aktivieren (wenn fokussiert)
Space	Button aktivieren (wenn fokussiert)

## Screen Reader Ankündigungen

Beim Öffnen:

"Dialog. [Modal-Titel]. [Modal-Beschreibung]. Dialog schließen, Button."

Beim Fokus auf Close-Button:

"Dialog schließen, Button."

Beim Schließen:

[Fokus kehrt zu Trigger zurück]  
"[Trigger-Element-Label], Button."

## Focus Indicators

WCAG 2.4.7: Fokussierte Elemente müssen visuell erkennbar sein.

```
.modal button:focus-visible {
  outline: 2px solid var(--color-hydrophon-blau-300);
  outline-offset: 2px;
}
```

## Color Contrast

WCAG 1.4.3: Text muss mindestens 4.5:1 Kontrast haben.

- Modal-Titel (neutral.900 auf white): 21:1 ✓
- Modal-Beschreibung (neutral.600 auf white): 7:1 ✓
- Button-Text: Siehe Button-Komponente (bereits WCAG-konform)

## Bestätigungs-Dialoge

Spezielle Modal-Variante für destruktive Aktionen.

### Verwendung

Wann:

- Löschen von Daten
- Verwerfen ungespeicherter Änderungen
- Abbrechen laufender Prozesse
- Andere irreversible Aktionen

**Warum:** Versehentliche destruktive Aktionen sind im B2B-Umfeld besonders kritisch. Ein gelöschter Auftrag oder verworfene Konfiguration kann Arbeitszeit kosten.

### Struktur

```
[X] [Warn-Icon] Artikel löschen
-----
Möchten Sie den Artikel
"Hydrophon Serie A" wirklich
löschen?
-----
Diese Aktion kann nicht
rückgängig gemacht werden.
-----
[Abbrechen] [Löschen]
```

## Best Practices

### 1. Explizite "Abbrechen"-Button ist Pflicht

Destruktive Aktion MUSS eine klare, sichere Alternative haben.

```
<div class="modal__footer">
  <button class="button button--secondary">Abbrechen</button>
  <button class="button button--primary button--error">Löschen</button>
</div>
```

### 2. Destructive Action verwendet Error-Farbe

Button für destruktive Aktion in Rot (color.error):

```
.button--error {
  background-color: var(--color-error-500);
  color: white;
}

.button--error:hover {
  background-color: var(--color-error-600);
}
```

### 3. Klare, spezifische Actionsbezeichnung

✗ Schlecht: "OK" / "Ja" ✓ Gut: "Löschen" / "Verwerfen" / "Abbrechen"

### 4. Kontext in Beschreibung

Beschreibung sollte benennen, WAS gelöscht wird:

✗ Schlecht: "Sind Sie sicher?" ✓ Gut: "Möchten Sie den Artikel 'Hydrophon Serie A' wirklich löschen?"

### 5. Konsequenzen erklären

Wenn irreversibel, explizit kommunizieren:

Diese Aktion kann nicht rückgängig gemacht werden.  
Alle zugehörigen Daten gehen verloren.

## Accessibility für Bestätigungs-Dialoge

Screenreader-Unterstützung:

```
<div role="alertdialog" aria-labelledby="confirm-title" aria-describedby="confirm-desc">
  <h2 id="confirm-title">Artikel löschen</h2>
  <p id="confirm-desc">Möchten Sie den Artikel "Hydrophon Serie A" wirklich löschen? Diese Aktion ist irreversibel.</p>
</div>
```

`role="alertdialog" vs role="dialog" :`

- `alertdialog` für kritische Bestätigungen (wird als Alert angekündigt)
- `dialog` für Standard-Modals

## Formular-Dialoge

Modals mit Formulareingaben erfordern besondere UX-Patterns.

### Best Practices

#### 1. Validierung vor Schließen

Verhindern, dass Modal mit ungültigen Daten geschlossen wird:

```
function handleSaveButton() {
  if (!validateForm()) {
    // Fehler inline anzeigen, Modal bleibt offen
    return;
  }

  saveData();
  closeModal();
}
```

#### 2. Ungespeicherte Änderungen warnen

Wenn Nutzer versucht, Modal mit ungespeicherten Daten zu schließen:

```
function handleCloseButton() {
  if (hasUnsavedChanges) {
    const confirmClose = confirm('Änderungen verwerfen?');
    if (!confirmClose) return;
  }

  closeModal();
}
```

#### Alternative: Auto-Save

Bei unkritischen Formularen automatisch speichern:

```
// Auto-Save bei jedem Input-Change (mit Debounce)
let saveTimeout;
InputElement.addEventListener('input', () => {
  clearTimeout(saveTimeout);
  saveTimeout = setTimeout(() => {
    autoSaveData();
  }, 1000);
});
```

#### 3. Fokus auf erstes Eingabefeld

Beim Öffnen eines Formular-Modals Fokus auf erstes Input setzen (nicht Close-Button):

```

function openFormModal() {
  showModal();

  const firstInput = modalElement.querySelector('input:not([disabled])');
  if (firstInput) {
    firstInput.focus();
  }
}

```

#### 4. Enter-Taste für Submit

Standard-Form-Submit per Enter-Taste ermöglichen:

```

<form onsubmit="handleSubmit(event)">
  <input type="text" name="name" />
  <button type="submit">Speichern</button>
</form>

```

#### 5. Fehler inline anzeigen

Validierungsfehler direkt unter fehlerhaftem Feld (nicht Modal schließen):

```

<div class="form-field">
  <label for="email">E-Mail</label>
  <input
    type="email"
    id="email"
    aria-describedby="email-error"
    aria-invalid="true"
  />
  <span id="email-error" class="form-error">
    Bitte gültige E-Mail-Adresse eingeben.
  </span>
</div>

```

---

## Technische Referenz

### Empfohlene Bibliothek: Radix UI Dialog

Warum Radix UI:

- **Vollständige Accessibility** — WAI-ARIA Dialog Pattern implementiert
- **Focus-Management** — Automatischer Focus-Trap und Fokus-Rückkehr
- **ESC-Taste** — Built-in ESC-Key-Handling
- **Portal-Rendering** — Automatisches Rendering außerhalb DOM-Hierarchie
- **Headless** — Bring-your-own-Styles (passt zu Token-System)

Installation:

```
npm install @radix-ui/react-dialog
```

**Basis-Implementierung:**

```

import * as Dialog from '@radix-ui/react-dialog';

interface ModalProps {
  open: boolean;
  onOpenChange: (open: boolean) => void;
  title: string;
  description?: string;
  size?: 'sm' | 'md' | 'lg';
  children: React.ReactNode;
}

export function Modal({
  open,
  onOpenChange,
  title,
  description,
  size = 'md',
  children
}: ModalProps) {
  return (
    <Dialog.Root open={open} onOpenChange={onOpenChange}>
      <Dialog.Portal>
        {/* Overlay (Backdrop) */}
        <Dialog.Overlay className="modal-overlay" />

        {/* Modal-Container */}
        <Dialog.Content className={`modal modal--${size}`}>
          {/* Header */}
          <div className="modal__header">
            <Dialog.Title className="modal__title">
              {title}
            </Dialog.Title>
            <Dialog.Close className="modal__close" aria-label="Dialog schließen">
              <svg><!-- X-Icon --></svg>
            </Dialog.Close>
          </div>

          {/* Body */}
          <div className="modal__body">
            {description && (
              <Dialog.Description className="modal__description">
                {description}
              </Dialog.Description>
            )}
            {children}
          </div>
        </Dialog.Content>
      </Dialog.Portal>
    </Dialog.Root>
  );
}

```

CSS für Radix Modal:

```
/* Overlay */
.modal-overlay {
  position: fixed;
  inset: 0;
  background-color: var(--modal-overlay-background);
  animation: fadeIn 200ms ease-out;
}

/* Modal-Container */
.modal {
  position: fixed;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background-color: var(--modal-container-background);
  border-radius: var(--modal-container-border-radius);
  padding: var(--modal-container-padding);
  box-shadow: var(--shadow-xl);
  max-height: 90vh;
  overflow-y: auto;
  animation: modalEnter 200ms ease-out;
}

.modal--sm {
  width: 90%;
  max-width: var(--modal-container-max-width-sm);
}

.modal--md {
  width: 90%;
  max-width: var(--modal-container-max-width-md);
}

.modal--lg {
  width: 90%;
  max-width: var(--modal-container-max-width-lg);
}

/* Header */
.modal__header {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
  margin-bottom: var(--modal-title-margin-bottom);
}

.modal__title {
  font-size: var(--modal-title-font-size);
  font-weight: var(--modal-title-font-weight);
  color: var(--modal-title-color);
  margin: 0;
}

.modal__close {
  width: var(--modal-close-button-size);
  height: var(--modal-close-button-size);
  padding: 0;
}
```

```

border: none;
background: transparent;
color: var(--modal-close-button-color);
cursor: pointer;
border-radius: var(--border-radius-md);
transition: color 150ms, background-color 150ms;
}

.modal__close:hover {
  color: var(--modal-close-button-hover-color);
  background-color: var(--color-neutral-100);
}

.modal__close:focus-visible {
  outline: 2px solid var(--color-hydrophon-blau-300);
  outline-offset: 2px;
}

/* Body */
.modal__body {
  color: var(--modal-description-color);
}

.modal__description {
  font-size: var(--modal-description-font-size);
  margin-bottom: var(--modal-description-margin-bottom);
}

/* Animationen */
@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}

@keyframes modalEnter {
  from {
    opacity: 0;
    transform: translate(-50%, -50%) scale(0.95);
  }
  to {
    opacity: 1;
    transform: translate(-50%, -50%) scale(1);
  }
}

/* Reduced Motion */
@media (prefers-reduced-motion: reduce) {
  .modal-overlay,
  .modal {
    animation: none;
  }
}

```

Verwendung:

```

function MyComponent() {
  const [isOpen, setIsOpen] = useState(false);

  return (
    <>
      <button onClick={() => setIsOpen(true)}>
        Modal öffnen
      </button>

      <Modal
        open={isOpen}
        onOpenChange={setIsOpen}
        title="Artikel löschen?"
        description="Diese Aktion kann nicht rückgängig gemacht werden."
        size="sm"
      >
        <div className="modal__footer">
          <button
            className="button button--secondary"
            onClick={() => setIsOpen(false)}
          >
            Abbrechen
          </button>
          <button
            className="button button--primary button--error"
            onClick={handleDelete}
          >
            Löschen
          </button>
        </div>
      </Modal>
    </>
  );
}

```

## Portal-Rendering

**Problem:** Modal befindet sich tief in DOM-Hierarchie, aber soll über allem erscheinen.

**Lösung:** Portal rendert Modal-Element außerhalb der normalen DOM-Struktur (üblicherweise direkt unter `<body>`).

**React Portal:**

```

import { createPortal } from 'react-dom';

function Modal({ children }) {
  return createPortal(
    <div className="modal-overlay">
      <div className="modal">
        {children}
      </div>
    </div>,
    document.body
  );
}

```

**Radix UI Dialog:** Portal ist bereits eingebaut ( `<Dialog.Portal>` ).

---

## Anti-Patterns

### ✗ Modal-Overuse

**Problem:** Jede Aktion öffnet ein Modal — Nutzer fühlen sich unterbrochen

**Beispiele:**

- Login in Modal statt dedizierte Seite
- Kontaktformular in Modal statt eigene Seite
- Produktdetails in Modal statt Detailseite

**Alternative:** Modals nur für kritische, kurze Interaktionen. Alles andere auf dedizierte Seiten.

### ✗ Backdrop-Click-Dismissal

**Problem:** Nutzer klickt versehentlich auf Backdrop, Modal schließt, Daten verloren

**Warum verbreitet:** Viele UI-Bibliotheken aktivieren dies standardmäßig

**Alternative:** Nur Close-Button + ESC-Taste als Schließ-Mechanismen

### ✗ Kritische Informationen in Modal

**Problem:** Wichtige Infos in Modal versteckt — schwer wiederzufinden

**Beispiele:**

- AGB in Modal statt dedizierte Seite
- Produktspezifikationen in Modal
- Hilfetext in Modal statt Inline

**Alternative:** Modals nur für temporäre Inhalte. Permanente Infos auf eigene Seiten.

### ✗ Nested Modals (Modal über Modal)

**Problem:** Verwirrend, schwer zu navigieren, Accessibility-Albtraum

**Beispiel:**

Modal 1: "Artikel bearbeiten"  
 → Nutzer klickt "Kategorie hinzufügen"  
 → Modal 2 öffnet über Modal 1  
 → Fokus-Management bricht  
 → ESC schließt welches Modal?

**Alternative:** Workflow umstrukturieren — Multi-Step-Forms, Breadcrumb-Navigation

### ✗ Fehlende ESC-Taste-Unterstützung

**Problem:** WCAG 2.1.2-Verletzung — Keyboard-Nutzer gefangen

**Alternative:** IMMER ESC-Taste implementieren (außer bei kritischen Aktionen mit Bestätigung)

### ✗ Kein Focus-Trap

**Problem:** Tab-Taste navigiert hinter Modal — verwirrend und nicht WCAG-konform

**Alternative:** Focus-Trap implementieren (siehe "Focus-Management" Sektion)

### ✗ Unspezifische Button-Labels

**Problem:** "OK" / "Cancel" sagen nichts über Aktion aus

**Alternative:** Spezifische Labels — "Löschen", "Speichern", "Abbrechen"

## Verwandte Komponenten

- [Tooltips](#) — Kontextuelle Hilfe für Icons und Buttons
- [Toasts](#) — Nicht-blockierende System-Benachrichtigungen
- [Mobile Drawer](#) — Alternative zu Modal für Navigation

## Changelog

Version	Datum	Änderungen
1.0	2026-01-29	Initiale Dokumentation (Modal-Dialoge)

# Tooltip

---

Tooltips sind kurze, kontextuelle Beschriftungen, die beim Hovern über ein Element erscheinen. Sie liefern zusätzliche Informationen zu Icons, Buttons oder abgeschnittenem Text.

**Anforderungen:** TTP-01 (Hover-Verhalten), TTP-02 (Positionierung), TTP-03 (Barrierefreiheit WCAG 1.4.13)

---

## Übersicht

Tooltips ergänzen die Benutzeroberfläche mit prägnanten Beschriftungen, die bei Bedarf sichtbar werden. Sie sind ideal für:

- **Icon-Buttons** — Benennung der Funktion (z.B. "Loeschen", "Bearbeiten")
- **Verkürzter Text** — Vollständiger Text bei abgeschnittenen Überschriften
- **Zusatzinformationen** — Ergänzende Details zu Formularfeldern

**Inhaltsbeschränkung:** 1-5 Wörter pro Tooltip. Tooltips sind keine Hilfetexte oder Erklärungen – nur kurze Labels.

**Beispiele guter Tooltip-Inhalte:**

- "Element loeschen"
- "Als PDF exportieren"
- "Ansicht aktualisieren"
- "In Zwischenablage kopieren"

**Was NICHT in Tooltips gehört:**

- Lange Erklärungen oder Beschreibungen
  - Kritische Informationen (nicht auf Touch-Geräten verfügbar)
  - Formular-Validierungsfehler (inline anzeigen)
  - Aktionsbestätigungen (Toast verwenden)
- 

## Verwendung

### Wann Tooltips verwenden

**Icon-Buttons ohne Text:** Icons allein können mehrdeutig sein. Ein Tooltip stellt sicher, dass die Funktion klar ist.

```
<button aria-label="Element loeschen">
  <TrashIcon />
  <!-- Tooltip: "Element loeschen" -->
</button>
```

**Abgeschnittener Text:** Wenn Text aus Platzgründen verkürzt wird (Ellipsis), zeigt der Tooltip den vollständigen Inhalt.

```
<div class="truncated-text" title="Sehr langer Produktnname der nicht in eine Zeile passt">
  Sehr langer Produktnname...
</div>
```

**Ergänzende Informationen:** Zusätzlicher Kontext für Formularfelder oder Einstellungen.

```
<label>
  Timeout-Dauer
  <InfoIcon aria-label="Weitere Informationen" />
  <!-- Tooltip: "In Sekunden" -->
</label>
```

## Wann KEINE Tooltips verwenden

**Kritische Informationen:** Tooltips sind auf Touch-Geräten schwer zugänglich. Kritische Infos müssen immer sichtbar oder über andere Mechanismen verfügbar sein.

**Mobile als Primärziel:** Touch-Geräte haben kein Hover-Ereignis. Tooltips funktionieren dort nur über Fokus (Tippen auf fokussierbares Element).

**Deaktivierte Elemente:** Deaktivierte Buttons empfangen kein Hover- oder Focus-Ereignis. Alternative: Wrapper-Element mit Tooltip oder Inline-Erklärung.

```
<!-- Anti-Pattern -->
<button disabled title="Nicht verfügbar">Speichern</button>

<!-- Besser -->
<div class="tooltip-wrapper" aria-label="Funktion nicht verfügbar: Formular unvollständig">
  <button disabled>Speichern</button>
</div>
```

---

## Design-Tokens

Alle Tooltip-Eigenschaften sind als Design-Token definiert und in `design-system/tokens/feedback.json` verfügbar.

## Tooltip-Token

Token	Wert	CSS-Variable	Verwendung
tooltip.delay	300ms	--tooltip-delay	Verzögerung vor dem Anzeigen
tooltip.background	neutral.900	--tooltip-background	Dunkler Hintergrund für hohen Kontrast
tooltip.color	neutral.white	--tooltip-color	Weisser Text auf dunklem Hintergrund
tooltip.font-size	typography.body.sm.fontSize	--tooltip-font-size	Kleine Schriftgröße (14px)
tooltip.padding.x	spacing.2	--tooltip-padding-x	Horizontaler Innenabstand (8px)
tooltip.padding.y	spacing.1	--tooltip-padding-y	Vertikaler Innenabstand (4px)
tooltip.border-radius	radius.sm	--tooltip-border-radius	Leichte Rundung (2px)
tooltip.max-width	200px	--tooltip-max-width	Maximale Breite vor Umbruch
tooltip.arrow.size	6px	--tooltip-arrow-size	Große des Pfeil-Indikators
tooltip.offset	5px	--tooltip-offset	Abstand zum Trigger-Element
tooltip.z-index	1100	--tooltip-z-index	Positionierung über anderen Inhalten
tooltip.animation.duration	150ms	--tooltip-animation-duration	Ein-/Ausblend-Animation

CSS-Anwendung:

```
.tooltip {
  background-color: var(--tooltip-background);
  color: var(--tooltip-color);
  font-size: var(--tooltip-font-size);
  padding: var(--tooltip-padding-y) var(--tooltip-padding-x);
  border-radius: var(--tooltip-border-radius);
  max-width: var(--tooltip-max-width);
  z-index: var(--tooltip-z-index);
}

.tooltip-arrow {
  width: var(--tooltip-arrow-size);
  height: var(--tooltip-arrow-size);
}
```

## Verhalten

### Hover-Trigger mit Verzögerung

Tooltips erscheinen nach **300ms Hover-Verzögerung**. Diese kurze Pause verhindert, dass Tooltips bei versehentlichem Überfahren aufblitzen.

#### Timing-Logik:

1. Mauszeiger bewegt sich über Trigger-Element
2. 300ms Wartezeit
3. Tooltip erscheint mit 150ms Fade-in
4. Mauszeiger verlässt Trigger-Element
5. Tooltip verschwindet mit 150ms Fade-out

#### Warum 300ms?

- Zu kurz (<200ms): Tooltips erscheinen bei schneller Mausbewegung ungewollt (Flackern)
- Zu lang (>500ms): Nutzer warten zu lange, wirkt träge
- 300ms: Industriestandard, optimal für Desktop-Nutzung

## Keyboard-Fokus

Tooltips müssen auch über Tastatur-Navigation erreichbar sein (WCAG 2.1.1 Keyboard).

#### Fokus-Verhalten:

- **Tab-Navigation:** Tooltip erscheint sofort bei Focus (keine Verzögerung)
- **ESC-Taste:** Schließt Tooltip
- **Tab weiter:** Tooltip schließt automatisch

```
<button aria-label="Element löschen" aria-describedby="delete-tooltip">
  <TrashIcon />
</button>
<div role="tooltip" id="delete-tooltip" hidden>
  Element löschen
</div>
```

## Positionierung

### Bevorzugte Position: Oben

Tooltips werden standardmäßig **oberhalb** des Trigger-Elements positioniert. Diese Position stört den Inhalt darunter nicht und folgt dem natürlichen Lesefluss (von oben nach unten).

#### Position-Priorität:

1. **Top** (Standard) — Oberhalb des Elements
2. **Bottom** — Wenn oben kein Platz (Viewport-Rand)
3. **Left/Right** — Wenn vertikal kein Platz

## Smart Positioning (Collision Detection)

Tooltips müssen am Viewport-Rand automatisch umpositioniert werden, um Abschneiden zu vermeiden.

### Empfohlene Bibliothek: Floating UI

Floating UI (ehemals Popper.js) ist der moderne Standard für intelligente Positionierung:

- **Automatisches Flipping:** Wechselt die Seite bei Platzmangel
- **Shift-Verhalten:** Verschiebt Tooltip horizontal bei seitlichem Abschneiden
- **Collision Padding:** Mindestabstand zum Viewport-Rand (10px)

Floating UI Integration:

```
import { computePosition, flip, shift, offset, arrow } from '@floating-ui/dom';

const triggerElement = document.querySelector('.trigger');
const tooltipElement = document.querySelector('.tooltip');

computePosition(triggerElement, tooltipElement, {
  placement: 'top',
  middleware: [
    offset(5), // 5px Abstand zum Trigger
    flip(), // Flip zu bottom/left/right bei Platzmangel
    shift({ padding: 10 }), // 10px Mindestabstand zum Viewport
    arrow({ element: arrowElement }) // Pfeil-Element
  ]
}).then(({ x, y, placement, middlewareData }) => {
  Object.assign(tooltipElement.style, {
    left: `${x}px`,
    top: `${y}px`,
  });
}

// Pfeil-Position basierend auf middlewareData.arrow
const arrowX = middlewareData.arrow.x;
const arrowY = middlewareData.arrow.y;
Object.assign(arrowElement.style, {
  left: arrowX != null ? `${arrowX}px` : '',
  top: arrowY != null ? `${arrowY}px` : '',
});
```

## Pfeil-Indikator

Ein kleiner Pfeil (6px) zeigt vom Tooltip zum Trigger-Element und verdeutlicht die Beziehung.

**Pfeil-Rendering (CSS):**

```
.tooltip-arrow {
  width: var(--tooltip-arrow-size);
  height: var(--tooltip-arrow-size);
  background-color: var(--tooltip-background);
  transform: rotate(45deg);
  position: absolute;
}

.tooltip[data-placement^="top"] .tooltip-arrow {
  bottom: calc(var(--tooltip-arrow-size) / -2);
}

.tooltip[data-placement^="bottom"] .tooltip-arrow {
  top: calc(var(--tooltip-arrow-size) / -2);
}
```

## Barrierefreiheit (WCAG 1.4.13)

WCAG 1.4.13 Content on Hover or Focus definiert drei Anforderungen für Hover-Inhalte wie Tooltips:

### 1. Dismissible (Schließbar)

Nutzer müssen Tooltips ohne Mausbewegung schließen können.

**Lösung:** ESC-Taste schließt Tooltip.

```
document.addEventListener('keydown', (event) => {
  if (event.key === 'Escape') {
    closeTooltip();
  }
});
```

### 2. Hoverable (Hover-fähig)

Der Tooltip-Inhalt selbst muss hoverbar sein. Nutzer müssen mit der Maus über den Tooltip fahren können, ohne dass er verschwindet.

**Problem:** Wenn Tooltip nur beim Hover des Triggers sichtbar ist, verschwindet er beim Verlassen des Triggers.

**Lösung:** Tooltip bleibt sichtbar, solange Maus über Trigger ODER Tooltip ist.

```

let isHoveringTrigger = false;
let isHoveringTooltip = false;

triggerElement.addEventListener('mouseenter', () => {
  isHoveringTrigger = true;
  showTooltip();
});

triggerElement.addEventListener('mouseleave', () => {
  isHoveringTrigger = false;
  setTimeout(() => {
    if (!isHoveringTooltip) {
      hideTooltip();
    }
  }, 100);
});

tooltipElement.addEventListener('mouseenter', () => {
  isHoveringTooltip = true;
});

tooltipElement.addEventListener('mouseleave', () => {
  isHoveringTooltip = false;
  if (!isHoveringTrigger) {
    hideTooltip();
  }
});

```

### 3. Persistent (Bestaendig)

Tooltip bleibt sichtbar, bis Nutzer den Hover beendet oder ESC drückt. Kein automatisches Ausblenden nach Zeit.

**Umsetzung:** Keine Auto-Dismisss-Timer für Tooltips (im Gegensatz zu Toasts).

#### ARIA-Attribute

Für Screenreader-Unterstützung:

```

<!-- Methode 1: aria-label auf Trigger (einfach) -->
<button aria-label="Element loeschen">
  <TrashIcon />
</button>

<!-- Methode 2: aria-describedby + role="tooltip" (detailliert) -->
<button aria-describedby="delete-tooltip-id">
  <TrashIcon />
</button>
<div role="tooltip" id="delete-tooltip-id" hidden>
  Element loeschen
</div>

```

**Empfehlung:** Methode 1 für einfache Labels, Methode 2 für komplexere Tooltips mit dynamischem Inhalt.

## Code-Beispiele

### Radix UI Tooltip (empfohlen)

Radix UI liefert eine barrierefreie Tooltip-Primitive, die alle WCAG-Anforderungen erfüllt.

Installation:

```
npm install @radix-ui/react-tooltip
```

Verwendung:

```
import * as Tooltip from '@radix-ui/react-tooltip';

export function IconButton() {
  return (
    <Tooltip.Provider delayDuration={300}>
      <Tooltip.Root>
        <Tooltip.Trigger asChild>
          <button className="icon-button" aria-label="Element loeschen">
            <TrashIcon />
          </button>
        </Tooltip.Trigger>
        <Tooltip.Portal>
          <Tooltip.Content
            className="tooltip"
            sideOffset={5}
            side="top"
            collisionPadding={10}
            avoidCollisions={true}
          >
            Element loeschen
            <Tooltip.Arrow className="tooltip-arrow" />
          </Tooltip.Content>
        </Tooltip.Portal>
      </Tooltip.Root>
    </Tooltip.Provider>
  );
}
```

Radix UI Features:

- 300ms Delay konfigurierbar ( `delayDuration` )
- Smart Positioning ( `side` , `avoidCollisions` , `collisionPadding` )
- WCAG 1.4.13 konform (hoverable, dismissible, persistent)
- Keyboard-Zugriff automatisch (Focus zeigt Tooltip)
- ESC-Taste schließt Tooltip
- Portal-Rendering (Z-Index-Probleme vermeiden)

Styling:

```

.tooltip {
  background-color: var(--tooltip-background);
  color: var(--tooltip-color);
  font-size: var(--tooltip-font-size);
  padding: var(--tooltip-padding-y) var(--tooltip-padding-x);
  border-radius: var(--tooltip-border-radius);
  max-width: var(--tooltip-max-width);
  z-index: var(--tooltip-z-index);
  animation: fadeIn var(--tooltip-animation-duration) ease-out;
}

.tooltip-arrow {
  fill: var(--tooltip-background);
}

@keyframes fadeIn {
  from { opacity: 0; transform: translateY(4px); }
  to { opacity: 1; transform: translateY(0); }
}

```

## Anti-Patterns

### ✗ Tooltips auf deaktivierten Elementen

**Problem:** Deaktivierte Elemente empfangen keine Mouse- oder Focus-Events.

Falsch:

```

<button disabled title="Funktion nicht verfuegbar">
  Speichern
</button>

```

Richtig: Wrapper-Element mit Tooltip:

```

<div class="tooltip-wrapper" aria-label="Speichern nicht moeglich: Formular unvollstaendig">
  <button disabled>Speichern</button>
</div>

```

### ✗ Kritische Informationen im Tooltip

**Problem:** Tooltips sind auf Touch-Geräten schwer zugänglich und werden leicht übersehen.

Falsch:

```
<button aria-label="Loeschen">
  <TrashIcon />
  <!-- Tooltip: "Achtung: Diese Aktion kann nicht rückgängig gemacht werden" -->
</button>
```

**Richtig:** Kritische Info inline oder in Bestätigungs-Modal:

```
<button aria-label="Loeschen" onClick={showConfirmDialog}>
  <TrashIcon />
</button>

<!-- Modal -->
<Dialog>
  <h2>Element löschen?</h2>
  <p>Diese Aktion kann nicht rückgängig gemacht werden.</p>
  <button>Löschen</button>
  <button>Abbrechen</button>
</Dialog>
```

## ✖ Lange Texte im Tooltip

**Problem:** Tooltips sind für kurze Labels gedacht, nicht für Erklärungen.

**Falsch:**

```
<InfoIcon aria-label="Weitere Informationen" />
<!-- Tooltip: "Die Timeout-Dauer bestimmt, wie lange das System auf eine Antwort wartet, bevor die"
```

**Richtig:** Kurzes Label + ausklappbare Hilfe:

```
<label>
  Timeout-Dauer
  <button aria-expanded="false" aria-controls="timeout-help">
    <InfoIcon />
    <!-- Tooltip: "Weitere Informationen" -->
  </button>
</label>
<div id="timeout-help" hidden>
  <p>Die Timeout-Dauer bestimmt, wie lange das System auf eine Antwort wartet...</p>
</div>
```

## ✖ Automatisches Ausblenden nach Zeit

**Problem:** Verstößt gegen WCAG 1.4.13 (Persistent). Nutzer müssen Zeit haben, Tooltip zu lesen.

**Falsch:**

```
setTimeout(() => {
  hideTooltip();
}, 3000); // Auto-Hide nach 3 Sekunden
```

**Richtig:** Tooltip bleibt sichtbar bis Hover/Focus endet:

```
// Kein Timer - nur bei mouseleave/blur schließen
triggerElement.addEventListener('mouseleave', hideTooltip);
triggerElement.addEventListener('blur', hideTooltip);
```

## Zusammenfassung

### Tooltip Best Practices:

- Kurz halten:** 1-5 Wörter, keine langen Erklärungen
- 300ms Delay:** Verhindert ungewolltes Aufblitzen
- Smart Positioning:** Floating UI für automatisches Flipping
- WCAG 1.4.13:** Hoverable, Dismissible, Persistent
- Keyboard-Zugriff:** Fokus zeigt Tooltip sofort
- Radix UI:** Empfohlene Bibliothek für barrierefreie Implementierung
- Nicht verwenden für:** Kritische Infos, lange Texte, deaktivierte Elemente, mobile Primärziele

### Siehe auch:

- [Toast-Benachrichtigungen](#) — Für Aktionsbestätigungen
- [Buttons](#) — Icon-Button-Muster
- [Forms](#) — Formular-Hilfe inline

---

Zuletzt aktualisiert: 2026-01-29

---

# Toast-Benachrichtigungen

---

Toast-Benachrichtigungen sind nicht-blockierende Rückmeldungen, die abgeschlossene Aktionen bestätigen, Systemzustände kommunizieren und wichtige Ereignisse melden.

**Anforderungen:** TST-01 (Toast-Varianten), TST-02 (Timing), TST-03 (Positionierung), TST-04 (Aktions-Buttons)

---

## Übersicht

Toasts erscheinen temporär am Bildschirmrand, um Nutzer\*innen über den Status von Aktionen zu informieren, ohne den aktuellen Workflow zu unterbrechen.

### Charakteristika:

- **Nicht-blockierend:** Nutzer können weiterarbeiten, während Toast sichtbar ist
- **Automatisches Schliessen:** Success/Info/Warning verschwinden nach festgelegter Zeit
- **Top-Right Position:** Konsistent mit Desktop-Anwendungen, stört Hauptinhalt nicht
- **Severity-basiert:** Vier Varianten (Success, Info, Warning, Error) mit unterschiedlichem Timing

### Verwendungszwecke:

- Aktionsbestätigungen ("Änderungen gespeichert")
- Systemmeldungen ("Neue Nachricht eingegangen")
- Warnungen ("Verbindung instabil")
- Fehlerbenachrichtigungen ("Fehler beim Speichern")

### Abgrenzung:

- **Nicht für Formular-Validierung:** Fehler direkt am Eingabefeld anzeigen (Phase 3)
  - **Nicht für kritische Entscheidungen:** Modal-Dialoge verwenden (Phase 5)
  - **Nicht für permanente Infos:** Inline-Meldungen oder Banner verwenden
- 

## Varianten

### Success (Erfolg)

**Verwendung:** Bestätigung erfolgreicher Aktionen.

#### Visuell:

- Hellgrüner Hintergrund (success.50: #f0fdf4)
- Grüner Rahmen (success.200: #bbf7d0)
- Grünes Icon (success.600: #16a34a)
- Dunkelgrüner Text (success.800: #166534)

**Timing:** Automatische Schließung nach **3 Sekunden**

#### Beispiele (Deutsch):

- "Änderungen gespeichert"
- "Element erfolgreich geloescht"

- "Datei hochgeladen"
- "Einstellungen aktualisiert"
- "Nachricht gesendet"

Mit Undo-Aktion:

```
toast.success('Element geloescht', {
  action: {
    label: 'Rueckgaengig',
    onClick: () => undoDelete(),
  },
});
```

## Info (Information)

**Verwendung:** Neutrale Informationen und System-Updates.

Visuell:

- Hellblauer Hintergrund (info.50: #eff6ff)
- Blauer Rahmen (info.200: #bfdbfe)
- Blaues Icon (info.600: #2563eb)
- Dunkelblauer Text (info.800: #1e40af)

**Timing:** Automatische Schließung nach **4 Sekunden**

**Beispiele (Deutsch):**

- "Neue Nachricht eingegangen"
- "Update verfuegbar"
- "Synchronisierung abgeschlossen"
- "Export wird vorbereitet"
- "Sitzung laeuft in 5 Minuten ab"

## Warning (Warnung)

**Verwendung:** Wichtige Hinweise, die Aufmerksamkeit erfordern, aber nicht kritisch sind.

Visuell:

- Hellgelber Hintergrund (warning.50: #ffffbe)
- Gelber Rahmen (warning.200: #fde68a)
- Oranges Icon (warning.600: #d97706)
- Dunkelorange Text (warning.800: #92400e)

**Timing:** Automatische Schließung nach **5 Sekunden**

**Beispiele (Deutsch):**

- "Verbindung instabil"
- "Speicherplatz wird knapp"
- "Ungespeicherte Aenderungen vorhanden"
- "Browser-Version veraltet"
- "Zeitlimit bald erreicht"

## Error (Fehler)

**Verwendung:** Fehler und kritische Probleme, die sofortige Aufmerksamkeit benötigen.

**Visuell:**

- Hellroter Hintergrund (error.50: #fef2f2)
- Roter Rahmen (error.200: #fecaca)
- Rotes Icon (error.600: #dc2626)
- Dunkelroter Text (error.800: #991b1b)

**Timing:** Bleibt sichtbar bis manuell geschlossen (duration: 0ms)

**Rationale:** Fehler erfordern Nutzeraktion oder -kenntnisnahme. Automatisches Schließen könnte dazu führen, dass kritische Fehler übersehen werden.

**Beispiele (Deutsch):**

- "Fehler beim Speichern. Bitte erneut versuchen."
- "Verbindung zum Server verloren"
- "Datei konnte nicht hochgeladen werden"
- "Ungültige Daten. Bitte prüfen."
- "Authentifizierung fehlgeschlagen"

Mit Retry-Aktion:

```
toast.error('Fehler beim Laden', {  
  action: {  
    label: 'Erneut versuchen',  
    onClick: () => retryLoad(),  
  },  
});
```

## Design-Token

Alle Toast-Eigenschaften sind als Design-Token definiert und in `design-system/tokens/feedback.json` verfügbar.

## Layout-Token

Token	Wert	CSS-Variable	Verwendung
toast.position.top	spacing.4	--toast-position-top	Abstand vom oberen Bildschirmrand (16px)
toast.position.right	spacing.4	--toast-position-right	Abstand vom rechten Bildschirmrand (16px)
toast.width	360px	--toast-width	Standardbreite für Toasts
toast.max-width	calc(100vw - 32px)	--toast-max-width	Responsive maximale Breite
toast.padding.x	spacing.4	--toast-padding-x	Horizontaler Innenabstand (16px)
toast.padding.y	spacing.3	--toast-padding-y	Vertikaler Innenabstand (12px)
toast.border-radius	radius.lg	--toast-border-radius	Eckenradius (8px)
toast.shadow	shadow.lg	--toast-shadow	Schatten für Elevation
toast.gap	spacing.3	--toast-gap	Abstand zwischen Icon und Text (12px)
toast.stack-gap	spacing.2	--toast-stack-gap	Abstand zwischen gestapelten Toasts (8px)
toast.max-visible	4	--toast-max-visible	Maximale Anzahl gleichzeitig sichtbar
toast.z-index	1200	--toast-z-index	Z-Index für Positionierung

## Timing-Token

Token	Wert	CSS-Variable	Verwendung
toast.duration.success	3000ms	--toast-duration-success	Auto-Dismiss nach 3 Sekunden
toast.duration.info	4000ms	--toast-duration-info	Auto-Dismiss nach 4 Sekunden
toast.duration.warning	5000ms	--toast-duration-warning	Auto-Dismiss nach 5 Sekunden
toast.duration.error	0ms	--toast-duration-error	Bleibt bis manuell geschlossen
toast.animation.enter-duration	200ms	--toast-animation-enter-duration	Einblend-Animation
toast.animation.exit-duration	150ms	--toast-animation-exit-duration	Ausblend-Animation

## Varianten-Token (Success)

Token	Wert	CSS-Variable	Verwendung
toast.variant.success.background	color.success.50	--toast-variant-success-background	Hellgrüner Hintergrund
toast.variant.success.border	color.success.200	--toast-variant-success-border	Grüner Rahmen
toast.variant.success.icon-color	color.success.600	--toast-variant-success-icon-color	Grünes Icon
toast.variant.success.text-color	color.success.800	--toast-variant-success-text-color	Dunkelgrüner Text

Info, Warning, Error folgen demselben Muster mit jeweiligen Farben.

## Aktions-Token

Token	Wert	CSS-Variable	Verwendung
toast.action.font-size	fontSize.sm	--toast-action-font-size	Schriftgröße für Aktions-Buttons (14px)
toast.action.font-weight	fontWeight.medium	--toast-action-font-weight	Schriftgewicht für Aktions-Buttons
toast.close-button.size	24px	--toast-close-button-size	Größe des Schließen-Buttons
toast.close-button.icon-size	16px	--toast-close-button-icon-size	Größe des X-Icons

## Positionierung

### Top-Right Corner

Toasts erscheinen in der **oberen rechten Ecke** des Viewports:

- 16px Abstand von oben (toast.position.top)
- 16px Abstand von rechts (toast.position.right)

Rationale:

- Industriestandard für Desktop-Anwendungen
- Stört den Hauptinhalt nicht (meist links/mittig)
- Konsistent mit natürlichem Blickverlauf (oben rechts = sekundäre Info)
- Zugänglich auf allen Bildschirmgrößen

CSS-Positionierung:

```
.toast-container {
  position: fixed;
  top: var(--toast-position-top);
  right: var(--toast-position-right);
  z-index: var(--toast-z-index);
  display: flex;
  flex-direction: column;
  gap: var(--toast-stack-gap);
}
```

## Timing

### Severity-basierte Auto-Dismisss

Die Anzeigedauer variiert je nach Schweregrad der Nachricht:

Variante	Dauer	Rationale
Success	3s	Bestätigung schnell erkannt, kurze Verweildauer ausreichend
Info	4s	Neutrale Info, etwas mehr Lesezeit
Warning	5s	Wichtiger Hinweis, mehr Zeit zum Verstehen
Error	$\infty$	Kritisch, bleibt bis Nutzer aktiv schließt

Warum Error nicht auto-dismissed wird:

- Fehler erfordern oft Nutzeraktion (z.B. "Erneut versuchen")
- Automatisches Schließen könnte zu übersehenden Fehlern führen
- Nutzer brauchen Zeit, Fehlertext zu lesen und zu verstehen
- WCAG-konform: Kritische Infos nicht zeitbeschränkt

**Pause-on-Hover:** Toasts pausieren den Auto-Dismiss-Timer bei Hover, damit Nutzer Zeit haben, den Text zu lesen oder Aktions-Buttons zu klicken.

## Stacking-Verhalten

### Vertikale Stapelung

Mehrere Toasts werden vertikal gestapelt:

- **Neueste oben:** Neue Toasts erscheinen oberhalb bestehender
- **8px Abstand:** Zwischen gestapelten Toasts (toast.stack-gap)
- **Maximale Anzahl:** 4 gleichzeitig sichtbar (toast.max-visible)

**Verhalten bei Limit-Überschreitung:** Wenn ein 5. Toast erscheint, wird der **älteste Toast automatisch geschlossen** (FIFO - First In, First Out).

**Beispiel-Reihenfolge:**



## Aktions-Buttons

Toasts können optionale Aktions-Buttons enthalten für häufige Interaktionen:

### Undo (Rückgängig)

**Verwendung:** Bei destruktiven Aktionen, die rückgängig gemacht werden können.

**Beispiele:**

- Element gelöscht → "Rückgängig" Button
- E-Mail archiviert → "Rückgängig" Button
- Benachrichtigung als gelesen markiert → "Rückgängig" Button

```
toast.success('Element geloescht', {
  duration: 3000,
  action: {
    label: 'Rueckgaengig',
    onClick: () => {
      restoreItem();
      toast.dismiss(toastId);
    },
  },
});
```

**Best Practice:** Undo-Toasts sollten länger sichtbar sein (5-10s statt 3s), damit Nutzer Zeit haben zu reagieren.

### Retry (Erneut versuchen)

**Verwendung:** Bei fehlgeschlagenen Aktionen, die wiederholt werden können.

**Beispiele:**

- Netzwerkfehler → "Erneut versuchen" Button
- Upload fehlgeschlagen → "Erneut versuchen" Button

- API-Aufruf gescheitert → "Erneut versuchen" Button

```
toast.error('Fehler beim Speichern. Bitte erneut versuchen.', {
  duration: Infinity, // Error-Toasts bleiben
  action: {
    label: 'Erneut versuchen',
    onClick: () => {
      retrySave();
    },
  },
});
```

## View Details (Details anzeigen)

**Verwendung:** Bei komplexen Ereignissen mit zusätzlichen Informationen.

**Beispiele:**

- "5 neue Benachrichtigungen" → "Details anzeigen"
- "Export abgeschlossen" → "Datei öffnen"
- "Update verfügbar" → "Mehr erfahren"

```
toast.info('Export abgeschlossen', {
  duration: 4000,
  action: {
    label: 'Datei öffnen',
    onClick: () => {
      window.open(downloadUrl, '_blank');
    },
  },
});
```

## Schließen-Button

Jeder Toast sollte einen **Close-Button (X)** haben:

- **24px × 24px** Größe (toast.close-button.size)
- **16px Icon** (toast.close-button.icon-size)
- **Top-right Position innerhalb des Toasts**

**Rationale:**

- Error-Toasts MÜSSEN schließbar sein (bleiben sonst ewig)
- Nutzer möchten manchmal Toasts manuell schließen
- Touch-Geräte: Tippen auf Close ist einfacher als Warten

## Barrierefreiheit

### ARIA Live Regions

Toasts müssen für Screenreader über ARIA Live Regions angekündigt werden:

**role="status"** für nicht-dringende Nachrichten:

- Success-Toasts
- Info-Toasts

```
<div role="status" aria-live="polite" class="toast toast--success">
  Änderungen gespeichert
</div>
```

**role="alert"** für dringende Nachrichten:

- Warning-Toasts
- Error-Toasts

```
<div role="alert" aria-live="assertive" class="toast toast--error">
  Fehler beim Speichern. Bitte erneut versuchen.
</div>
```

**Unterschied:**

- `aria-live="polite"` — Screenreader wartet, bis aktuelle Ausgabe endet
- `aria-live="assertive"` — Screenreader unterbricht und liest sofort

### Pause on Hover/Focus

Auto-Dismis-Timer muss pausieren, wenn:

- Maus über Toast (Hover)
- Toast oder dessen Button im Fokus (Keyboard-Navigation)

**Warum:** Nutzer brauchen Zeit, um:

- Text zu lesen
- Aktions-Button zu erreichen
- Screenreader-Ausgabe zu hören

```

let dismissTimer;

function showToast(message, duration) {
  const toast = createToastElement(message);

  dismissTimer = setTimeout(() => {
    dismissToast(toast);
  }, duration);

  toast.addEventListener('mouseenter', () => {
    clearTimeout(dismissTimer);
  });

  toast.addEventListener('mouseleave', () => {
    dismissTimer = setTimeout(() => {
      dismissToast(toast);
    }, duration);
  });

  toast.addEventListener('focusin', () => {
    clearTimeout(dismissTimer);
  });

  toast.addEventListener('focusout', () => {
    dismissTimer = setTimeout(() => {
      dismissToast(toast);
    }, duration);
  });
}

```

## Keyboard-Navigation

Toasts und deren Aktions-Buttons müssen per Tastatur erreichbar sein:

### Tab-Navigation:

- Fokus springt zu Close-Button und Aktions-Buttons
- Fokus-Ring sichtbar (WCAG 2.4.7)

### Enter/Space:

- Aktiviert fokussierten Button

### ESC-Taste:

- Schließt alle sichtbaren Toasts

```

document.addEventListener('keydown', (event) => {
  if (event.key === 'Escape') {
    dismissAllToasts();
  }
});

```

## Code-Beispiele

### Sonner (empfohlen)

Sonner ist die moderne Standard-Bibliothek für React Toast-Benachrichtigungen.

Installation:

```
npm install sonner
```

Setup im App-Root:

```
import { Toaster } from 'sonner';

export function App() {
  return (
    <>
      <Toaster
        position="top-right"
        visibleToasts={4}
        duration={3000}
        expand={false}
        richColors={false}
      />
      {/* Rest der App */}
    </>
  );
}
```

Verwendung in Komponenten:

```
import { toast } from 'sonner';

function SaveButton() {
  const handleSave = async () => {
    try {
      await saveData();
      toast.success('Änderungen gespeichert');
    } catch (error) {
      toast.error('Fehler beim Speichern. Bitte erneut versuchen.', {
        action: {
          label: 'Erneut versuchen',
          onClick: () => handleSave(),
        },
      });
    }
  };

  return <button onClick={handleSave}>Speichern</button>;
}
```

**Severity-basiertes Timing:**

```
// Success: 3 Sekunden
toast.success('Element geloescht', {
  duration: 3000,
  action: {
    label: 'Rueckgaengig',
    onClick: () => undoDelete(),
  },
});

// Info: 4 Sekunden
toast.info('Neue Nachricht eingegangen', {
  duration: 4000,
});

// Warning: 5 Sekunden
toast.warning('Verbindung instabil', {
  duration: 5000,
});

// Error: Bleibt bis geschlossen
toast.error('Fehler beim Laden', {
  duration: Infinity,
  action: {
    label: 'Erneut versuchen',
    onClick: () => retryLoad(),
  },
});
```

**Sonner Features:**

- Top-right Positionierung (konfigurierbar)
- Severity-basiertes Timing konfigurierbar
- Stacking mit max 4 sichtbar (visibleToasts)
- Pause-on-hover automatisch
- ARIA Live Regions automatisch
- Aktions-Buttons mit `action` prop
- Keine manuelle Setup-Logik nötig

**Styling mit Design-Token:**

```
[data-sonner-toaster] {
  --toast-width: var(--toast-width);
  --toast-padding-x: var(--toast-padding-x);
  --toast-padding-y: var(--toast-padding-y);
  --toast-border-radius: var(--toast-border-radius);
  --toast-shadow: var(--toast-shadow);
  --toast-gap: var(--toast-gap);
}

[data-sonner-toast][data-type="success"] {
  background-color: var(--toast-variant-success-background);
  border: 1px solid var(--toast-variant-success-border);
  color: var(--toast-variant-success-text-color);
}

[data-sonner-toast][data-type="error"] {
  background-color: var(--toast-variant-error-background);
  border: 1px solid var(--toast-variant-error-border);
  color: var(--toast-variant-error-text-color);
}
```

## Deutsche Nachricht-Beispiele

### Success-Nachrichten

- "Änderungen gespeichert"
- "Element erfolgreich gelöscht"
- "Datei hochgeladen"
- "Einstellungen aktualisiert"
- "Nachricht gesendet"
- "Benutzer hinzugefügt"
- "Passwort geändert"
- "Zahlungsmethode aktualisiert"

### Info-Nachrichten

- "Neue Nachricht eingegangen"
- "Update verfügbar"
- "Synchronisierung abgeschlossen"
- "Export wird vorbereitet"
- "Sitzung läuft in 5 Minuten ab"
- "5 neue Benachrichtigungen"
- "Download gestartet"
- "Seite wurde aktualisiert"

### Warning-Nachrichten

- "Verbindung instabil"
- "Speicherplatz wird knapp"

- "Ungespeicherte Änderungen vorhanden"
- "Browser-Version veraltet"
- "Zeitlimit bald erreicht"
- "Maximale Anzahl erreicht"
- "Langsame Internetverbindung"
- "Cache muss geleert werden"

## Error-Nachrichten

- "Fehler beim Speichern. Bitte erneut versuchen."
- "Verbindung zum Server verloren"
- "Datei konnte nicht hochgeladen werden"
- "Ungültige Daten. Bitte prüfen."
- "Authentifizierung fehlgeschlagen"
- "Zugriff verweigert"
- "Timeout: Vorgang abgebrochen"
- "Element nicht gefunden"

## Formulierungs-Richtlinien:

- **Klar und prägnant:** Maximal 1-2 Sätze
- **Handlungsorientiert:** Was ist passiert, was tun?
- **Freundlich:** "Bitte erneut versuchen" statt "Fehler"
- **Deutsch:** Sie-Form, höflich aber direkt
- **Keine Fachbegriffe:** "Verbindung verloren" statt "HTTP 503 Error"

## Anti-Patterns

### ✗ Formular-Validierung in Toasts

**Problem:** Fehler sollten inline am betroffenen Feld erscheinen, nicht in Toast-Benachrichtigungen.

Falsch:

```
// Formular absenden
if (!email.isValid) {
  toast.error('E-Mail-Adresse ungültig'); // ✗
}
```

Richtig:

```
<!-- Inline-Fehler -->
<input type="email" aria-invalid="true" aria-describedby="email-error" />
<span id="email-error" class="error">
  Bitte geben Sie eine gültige E-Mail-Adresse ein.
</span>
```

**Rationale:** Formular-Fehler müssen direkt am betroffenen Feld sichtbar sein, damit Nutzer wissen, was zu korrigieren ist.

## ✖ Auto-Dismissing Error-Toasts

**Problem:** Fehler verschwinden, bevor Nutzer sie lesen oder darauf reagieren können.

Falsch:

```
toast.error('Fehler beim Laden', {
  duration: 3000, // ✖ Verschwindet automatisch
});
```

Richtig:

```
toast.error('Fehler beim Laden', {
  duration: Infinity, // ✅ Bleibt bis manuell geschlossen
  action: {
    label: 'Erneut versuchen',
    onClick: () => retryLoad(),
  },
});
```

## ✖ Zu viele gleichzeitige Toasts

**Problem:** Mehr als 4 Toasts überfordern den Nutzer und überdecken Inhalte.

Falsch:

```
// Mehrere Aktionen hintereinander
items.forEach(item => {
  toast.success(`#${item.name} geloescht`); // ✖ 10+ Toasts
});
```

Richtig:

```
// Zusammengefasste Nachricht
toast.success(`#${items.length} Elemente geloescht`, {
  action: {
    label: 'Rueckgaengig',
    onClick: () => undoDeleteAll(items),
  },
});
```

## ✖ Toast-Spam bei schnellen Aktionen

**Problem:** Wiederholte Toasts bei schnell aufeinanderfolgenden Aktionen (z.B. "Like"-Button).

Falsch:

```
function handleLike() {
  toast.success('Beitrag geliked'); // ❌ Bei jedem Klick
}
```

Richtig:

```
// Optimistic UI ohne Toast
function handleLike() {
  setLiked(!liked); // Sofortiges visuelles Feedback
  // Kein Toast nötig - UI-Änderung ist ausreichend
}
```

## ❌ Technische Fehler-Nachrichten

Problem: Technischer Jargon verwirrt Nutzer.

Falsch:

```
toast.error('HTTP 500 Internal Server Error'); // ❌
```

Richtig:

```
toast.error('Serverfehler. Bitte versuchen Sie es spaeter erneut.', {
  action: {
    label: 'Erneut versuchen',
    onClick: () => retry(),
  },
});
```

## Zusammenfassung

Toast Best Practices:

- ✓ **Top-Right Position:** 16px Abstand von oben und rechts
- ✓ **Severity-basiertes Timing:** 3s/4s/5s/∞ je nach Wichtigkeit
- ✓ **Max 4 sichtbar:** Älteste verschwindet bei Überschreitung
- ✓ **Error bleibt:** Fehler nie auto-dismissing
- ✓ **Aktions-Buttons:** Undo, Retry, View Details wo sinnvoll
- ✓ **ARIA Live Regions:** role="status" (Success/Info), role="alert" (Warning/Error)
- ✓ **Pause-on-Hover:** Timer pausiert bei Hover/Focus
- ✓ **Sonner empfohlen:** Moderne Bibliothek mit allen Features
- ❌ **Nicht verwenden für:** Formular-Validierung, kritische Entscheidungen, permanente Infos, zu viele gleichzeitige Meldungen

Siehe auch:

- [Toolips](#) — Für kurze Labels bei Hover
  - [Modal-Dialoge](#) — Für kritische Bestätigungen
  - [Form Validation](#) — Inline-Fehler bei Formularen
- 

Zuletzt aktualisiert: 2026-01-29

---

# Ladezustände

---

Ladeindikatoren halten Nutzer während asynchroner Operationen bei Laune und vermitteln, dass das System arbeitet.

## Übersicht

Ladezustände sind ein kritischer Teil der User Experience. Sie zeigen Nutzern, dass ihre Aktion erfolgreich registriert wurde und das System arbeitet. Ohne Feedback entsteht schnell der Eindruck, die Anwendung sei abgestürzt oder die Aktion nicht erfolgreich gewesen.

### Zweck:

- Nutzer wissen, dass das System ihre Aktion verarbeitet
- Vermittelt gefühlte Geschwindigkeit durch sofortiges visuelles Feedback
- Verhindert Doppel-Klicks und wiederholte Aktionen
- Bietet Kontext über Fortschritt bei längeren Operationen

### Drei Arten von Ladeindikatoren:

1. **Spinner** – Unbestimmte kurze Aktionen (<3 Sekunden)
2. **Progress Bar** – Längere Aktionen mit bekannter Dauer (Uploads, Downloads)
3. **Skeleton Screens** – Content-Laden (Cards, Tabellen, Listen)

## Wann welchen Indikator verwenden

Die Wahl des richtigen Ladeindikators hängt von Kontext und erwarteter Dauer ab:

Kontext	Erwartete Dauer	Indikator	Beispiel
Button-Aktion	< 1 Sekunde	Spinner (delayed)	Formular absenden, Element löschen
API-Call	1-3 Sekunden	Spinner (immediate)	Daten nachladen, Suche
Upload/Download	> 3 Sekunden, bekannte Größe	Progress Bar (determinate)	Datei-Upload, Export
Verarbeitung	> 3 Sekunden, unbekannte Dauer	Progress Bar (indeterminate)	Bildverarbeitung, PDF-Generierung
Content-Laden	Beliebig	Skeleton Screen	Produktkarten, Tabellen, Listen

### Faustregel:

- **Spinner:** "Ich arbeite daran"
- **Progress Bar:** "Ich bin zu X% fertig" oder "Ich arbeite, weiß aber nicht wie lange"
- **Skeleton Screen:** "So wird der Content aussehen, gleich ist er da"

## Spinner

Kreisförmiger Ladeindikator für kurze, unbestimmte Wartezeiten.

## Größen

Spinner-Größen sind mit dem Icon-System aus Phase 2 abgestimmt:

Token	Größe	Verwendung
spinner.size.sm	16px	Kleine Buttons, Inline-Icons
spinner.size.md	24px	Standard-Buttons, Standard-Kontext
spinner.size.lg	32px	Große Buttons, Seitenbereiche
spinner.size.xl	48px	Vollbild-Lader, zentrale Seiten-Ladeindikator

## Farben

Token	Verwendung
spinner.color.primary	Hydrophon Blau – Standard auf weißem/hellem Hintergrund
spinner.color.light	Neutral 400 – Sekundärer Spinner, weniger Aufmerksamkeit
spinner.color.on-primary	Weiß – Spinner auf primärem Button (blauer Hintergrund)

Track Color: spinner.track-color (neutral.200) – Hintergrundkreis für Kontrast

## Verzögertes Erscheinen

**Problem:** Spinner, die sofort erscheinen und bei schnellen Operationen (<200ms) sofort wieder verschwinden, erzeugen störende Flashes.

**Lösung:** spinner.delay (200ms) – Spinner erscheint erst nach 200-300ms Wartezeit.

**Implementierung:**

```

function DelayedSpinner({ delay = 200, size = 'md' }) {
  const [show, setShow] = useState(false);

  useEffect(() => {
    const timer = setTimeout(() => setShow(true), delay);
    return () => clearTimeout(timer);
  }, [delay]);

  if (!show) return null;

  return (
    <svg
      className={`spinner spinner-${size}`}
      width={`var(--spinner-size-${size})`}
      height={`var(--spinner-size-${size})`}
      viewBox="0 0 24 24"
      xmlns="http://www.w3.org/2000/svg"
    >
      <circle
        cx="12"
        cy="12"
        r="10"
        stroke="var(--spinner-track-color)"
        strokeWidth="var(--spinner-stroke-width)"
        fill="none"
      />
      <circle
        cx="12"
        cy="12"
        r="10"
        stroke="var(--spinner-color-primary)"
        strokeWidth="var(--spinner-stroke-width)"
        fill="none"
        strokeDasharray="31.4 31.4"
        strokeDashoffset="0"
        strokeLinecap="round"
      />
      <animateTransform
        attributeName="transform"
        type="rotate"
        from="0 12 12"
        to="360 12 12"
        dur="var(--spinner-animation-duration)"
        repeatCount="indefinite"
      />
    </circle>
  </svg>
);
}

```

## CSS-Animation

Alternative: Reine CSS-Lösung mit `@keyframes` :

```
.spinner {
  animation: spin var(--spinner-animation-duration) var(--spinner-animation-timing) infinite;
}

@keyframes spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

## Barrierefreiheit

ARIA-Attribute:

```
<div role="status" aria-live="polite" aria-label="Wird geladen">
  <Spinner />
</div>
```

- `role="status"` – Screenreader erkennt Statusänderung
- `aria-live="polite"` – Ankündigung nach aktueller Ausgabe
- `aria-label="Wird geladen"` – Beschreibender Text für Screenreader

**Tastaturnutzer:** Spinner selbst ist nicht fokussierbar. Bei Button-Spinner sollte der Button disabled sein während Laden.

## Progress Bar

Balken-Indikator für längere Aktionen mit oder ohne bekannten Fortschritt.

### Determinate vs. Indeterminate

**Determinate:** Fortschritt ist bekannt (0-100%)

- Datei-Upload (Bytes hochgeladen / Gesamtgröße)
- Download
- Mehrstufige Formulare (Schritt X von Y)

**Indeterminate:** Fortschritt ist unbekannt

- Server-Verarbeitung unbekannter Dauer
- Komplexe Berechnungen
- Warteschlange ohne Zeit-Schätzung

### Native HTML Progress Element

**Empfehlung:** Verwende das native `<progress>` Element – Accessibility out-of-the-box.

**Determinate:**

```
<div class="progress-container">
  <label for="upload-progress">Datei wird hochgeladen</label>
  <progress
    id="upload-progress"
    value="45"
    max="100"
    aria-valuenow="45"
    aria-valuemin="0"
    aria-valuemax="100"
  >
    45%
  </progress>
  <span class="progress-label">45%</span>
</div>
```

Indeterminate:

```
<progress aria-label="Wird verarbeitet">Wird verarbeitet...</progress>
```

Kein **value** Attribut = Browser rendert indeterminate animation.

## Styling

```
.progress-container {
  display: flex;
  flex-direction: column;
  gap: var(--progress-label-gap);
}

progress {
  width: 100%;
  height: var(--progress-height);
  border-radius: var(--progress-border-radius);
  overflow: hidden;
  appearance: none;
  border: none;
}

/* Track (Hintergrund) */
progress::-webkit-progress-bar {
  background: var(--progress-track-background);
  border-radius: var(--progress-border-radius);
}

progress::-moz-progress-bar {
  background: var(--progress-track-background);
}

/* Bar (Fortschritt) */
progress::-webkit-progress-value {
  background: var(--progress-bar-background);
  border-radius: var(--progress-border-radius);
  transition: width 0.3s ease;
}

progress::-moz-progress-bar {
  background: var(--progress-bar-background);
  border-radius: var(--progress-border-radius);
}

/* Indeterminate Animation */
progress:indeterminate::-webkit-progress-bar {
  background: linear-gradient(
    90deg,
    var(--progress-track-background) 0%,
    var(--progress-bar-background) 50%,
    var(--progress-track-background) 100%
  );
  background-size: 200% 100%;
  animation: indeterminate var(--progress-indeterminate-animation-duration) linear infinite;
}

@keyframes indeterminate {
  from {
    background-position: 200% 0;
  }
  to {
    background-position: -200% 0;
  }
}
```

```

        }
    }

.progress-label {
    font-size: var(--progress-label-font-size);
    color: var(--progress-label-color);
    text-align: right;
}

```

## Erfolgsstatus

Bei Abschluss auf 100% kann die Bar-Farbe auf Erfolgsgrün wechseln:

```

function ProgressBar({ value, max = 100, label }) {
    const percentage = Math.round((value / max) * 100);
    const isComplete = percentage >= 100;

    return (
        <div className="progress-container">
            {label && <label>{label}</label>}
            <progress
                value={value}
                max={max}
                className={isComplete ? 'progress-success' : ''}
                aria-valuenow={value}
                aria-valuemin="0"
                aria-valuemax={max}
            >
                {percentage}%
            </progress>
            <span className="progress-label">{percentage}%</span>
        </div>
    );
}

```

```

progress.progress-success::-webkit-progress-value {
    background: var(--progress-bar-success);
}

progress.progress-success::-moz-progress-bar {
    background: var(--progress-bar-success);
}

```

## Barrierefreiheit

ARIA-Attribute für Determinate Progress:

- `aria-valuenow` – Aktueller Wert
- `aria-valuemin` – Minimum (0)
- `aria-valuemax` – Maximum (100)
- `aria-label` oder `<label>` – Beschreibung der Operation

### Screenreader-Ansage:

Bei Fortschrittsänderungen **nicht** bei jedem Prozentpunkt ankündigen (zu verbose). Entweder:

- Nur bei 25%, 50%, 75%, 100%
- Oder mit `aria-live="polite"` und debounced Updates

## Skeleton Screens

Content-Platzhalter, die das finale Layout vorwegnehmen.

### Was sind Skeleton Screens?

Skeleton Screens sind Platzhalter, die die Struktur des kommenden Contents nachbilden – graue Boxen in Form von Text, Bildern, Buttons. Sie vermitteln:

1. **Was kommt:** Nutzer sehen die Struktur, bevor Content da ist
2. **Kein Stillstand:** Seite fühlt sich schneller an (Progressive Loading)
3. **Kein Layout Shift:** Platzhalter matcht finale Größe exakt

**Wann verwenden:**

- Produktkarten-Grids
- Tabellen
- Listen (News, Blog-Posts)
- Detailseiten

**Wann NICHT verwenden:**

- Sehr schnelles Laden (<300ms) – Skeleton würde flashen
- Unvorhersehbare Content-Struktur
- Einfache Texte/Überschriften – einfacher Spinner reicht

## react-loading-skeleton Empfehlung

Library: [react-loading-skeleton](#)

Warum:

- Auto-sizing auf Content-Größe (kein manuelles Hardcoding)
- Eingebauter Shimmer-Effekt
- Einfachste API: `<Skeleton />` – fertig
- Themable für Design-System-Integration

Installation:

```
npm install react-loading-skeleton
```

Setup mit Design-System:

```
import Skeleton, { SkeletonTheme } from 'react-loading-skeleton';
import 'react-loading-skeleton/dist/skeleton.css';

function App() {
  return (
    <SkeletonTheme
      baseColor="var(--skeleton-base-color)"
      highlightColor="var(--skeleton-highlight-color)"
    >
      {/* App content */}
    </SkeletonTheme>
  );
}


```

## Skeleton-Patterns

Produktkarte Skeleton:

```

function ProductCardSkeleton() {
  return (
    <article className="card">
      {/* Image (1:1 Aspect Ratio) */}
      <Skeleton height={280} />

      <div className="card-content">
        {/* Produktname */}
        <Skeleton height={24} width="80%" style={{ marginTop: 16 }} />

        {/* Specs (3 Zeilen) */}
        <Skeleton count={3} height={16} style={{ marginTop: 8 }} />

        {/* CTA Button */}
        <Skeleton height={40} width={120} style={{ marginTop: 16 }} />
      </div>
    </article>
  );
}

function ProductGrid({ loading, products }) {
  if (loading) {
    return (
      <div className="product-grid">
        {Array(6)
          .fill(0)
          .map((_, i) =>
            <ProductCardSkeleton key={i} />
          ))}
      </div>
    );
  }

  return (
    <div className="product-grid">
      {products.map((product) => (
        <ProductCard key={product.id} product={product} />
      ))}
    </div>
  );
}

```

Tabelle Skeleton:

```
function TableSkeleton({ rows = 5, columns = 4 }) {
  return (
    <table className="table">
      <thead>
        <tr>
          {Array(columns)
            .fill(0)
            .map(_, i) => (
              <th key={i}>
                <Skeleton width="80%" />
              </th>
            ))}
        </tr>
      </thead>
      <tbody>
        {Array(rows)
          .fill(0)
          .map(_, rowIndex) => (
            <tr key={rowIndex}>
              {Array(columns)
                .fill(0)
                .map(_, colIndex) => (
                  <td key={colIndex}>
                    <Skeleton />
                  </td>
                ))}
            </tr>
          ))}
      </tbody>
    </table>
  );
}
```

## Shimmer-Animation

GPU-beschleunigt: Animate `transform: translateX()` statt `background-position` für 60fps.

```

@keyframes shimmer {
  0% {
    transform: translateX(-100%);
  }
  100% {
    transform: translateX(100%);
  }
}

.skeleton {
  position: relative;
  overflow: hidden;
  background: var(--skeleton-base-color);
  border-radius: var(--skeleton-border-radius);
}

.skeleton::after {
  content: '';
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: linear-gradient(
    90deg,
    transparent 0%,
    var(--skeleton-highlight-color) 50%,
    transparent 100%
  );
  animation: shimmer var(--skeleton-animation-duration) var(--skeleton-animation-timing) infinite;
}

```

## Layout-Match

**Wichtig:** Skeleton muss EXAKT das finale Layout matchen, sonst entsteht Layout Shift.

**Checkliste:**

- Gleiche Grid-Spalten/Struktur
- Gleiche Padding/Margins
- Gleiche Border-Radius
- Gleiche Höhen (oder aspect-ratio)
- Gleiche Schriftgröße (bei Text-Skeleton)

**Beispiel:** Produktkarte hat `aspect-ratio: 1/1` für Bild → Skeleton Image braucht gleiche Höhe:

```
<Skeleton height={280} /> /* Entspricht 280px card-image */
```

## Optimistic UI

Zeige Erfolg sofort, rollback bei Fehler.

## Was ist Optimistic UI?

Statt auf Server-Response zu warten, zeigt die UI **sofort** den Erfolg an – "optimistisch" annehmend, dass die Aktion erfolgreich sein wird.

### Beispiel:

1. User klickt "Like"
2. UI zeigt sofort "Liked" (Herz gefüllt, Counter +1)
3. API-Call im Hintergrund
4. **Erfolg:** Alles gut, kein Change nötig
5. **Fehler:** Rollback zu "Not Liked" + Error Toast

**Gefühlte Geschwindigkeit:** Instant Response vs. 200-500ms Wartezeit.

## Wann verwenden

### Gut geeignet:

- Like/Unlike (Social Features)
- Favorit hinzufügen/entfernen
- Artikel in Warenkorb legen
- Einfache Toggle-Aktionen
- Drafts auto-speichern

### NICHT geeignet:

- Destruktive Aktionen (Löschen, Konto schließen)
- Finanztransaktionen (Zahlung, Überweisung)
- Kritische Operationen (Veröffentlichung, Freigabe)
- Aktionen mit komplexen Seiteneffekten

**Faustregel:** Nur für Aktionen, die:

1. Sehr wahrscheinlich erfolgreich sind (>95%)
2. Einfach rückgängig zu machen sind
3. Keine kritischen Konsequenzen haben

## React useOptimistic Hook

React 19+ bietet `useOptimistic` Hook für optimistische Updates:

```

import { useOptimistic } from 'react';
import { toast } from 'sonner';

function LikeButton({ postId, initialLikes, initialLiked }) {
  const [optimisticState, setOptimisticState] = useOptimistic(
    { likes: initialLikes, liked: initialLiked },
    (current, newLiked) => ({
      likes: current.likes + (newLiked ? 1 : -1),
      liked: newLiked,
    })
  );

  async function handleLike() {
    const newLiked = !optimisticState.liked;

    // Sofortiges UI-Update (optimistisch)
    setOptimisticState(newLiked);

    try {
      // API-Call im Hintergrund
      await updateLike(postId, newLiked);
      // Erfolg: State ist bereits richtig
    } catch (error) {
      // Fehler: useOptimistic rollt automatisch zurück
      toast.error('Like konnte nicht gespeichert werden', {
        action: {
          label: 'Erneut versuchen',
          onClick: () => handleLike(),
        },
      });
    }
  }

  return (
    <button onClick={handleLike} className={optimisticState.liked ? 'liked' : ''}>
      {optimisticState.likes}
    </button>
  );
}

```

#### Vorteile useOptimistic:

- Automatischer Rollback bei Fehler
- Keine manuellen State-Flags (isOptimistic, isPending)
- Race-Condition-sicher
- Server State bleibt Source of Truth

#### Rollback mit Toast

Bei Fehler: Rollback + Error Toast mit Retry

```
catch (error) {
  // useOptimistic rollt zurück
  toast.error('Aktion fehlgeschlagen', {
    duration: 5000,
    action: [
      {label: 'Erneut versuchen',
       onClick: () => handleLike()},
    ],
  });
}
```

#### Nutzer-Kommunikation:

- Toast erscheint top-right (nicht blockierend)
- Fehlertext erklärt, was schiefging
- Retry-Button bietet einfache Wiederholung
- 5s duration (länger als Success, kürzer als manual-only)

## Fehlerbehandlung

Wenn Laden fehlschlägt: Error Toast + Retry.

### Error State Pattern

Komponente hat drei States:

1. **Loading:** Spinner/Skeleton wird angezeigt
2. **Success:** Content wird angezeigt
3. **Error:** Error-Message + Retry-Button

Implementierung:

```

function ProductList() {
  const [products, setProducts] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  async function loadProducts() {
    setLoading(true);
    setError(null);

    try {
      const data = await fetchProducts();
      setProducts(data);
    } catch (err) {
      setError(err.message);
      toast.error('Produkte konnten nicht geladen werden', {
        duration: Infinity,
        action: {
          label: 'Erneut versuchen',
          onClick: () => loadProducts(),
        },
      });
    } finally {
      setLoading(false);
    }
  }

  useEffect(() => {
    loadProducts();
  }, []);
}

if (loading) {
  return (
    <div className="product-grid">
      {Array(6)
        .fill(0)
        .map((_, i) => (
          <ProductCardSkeleton key={i} />
        ))}
    </div>
  );
}

if (error) {
  return (
    <div className="error-state">
      <p>Fehler beim Laden der Produkte</p>
      <button onClick={loadProducts}>Erneut versuchen</button>
    </div>
  );
}

return (
  <div className="product-grid">
    {products.map((product) => (
      <ProductCard key={product.id} product={product} />
    ))}
  </div>
)
}

```

```
    );
}
```

## Non-Blocking Errors

**Toast statt Inline:** Bei Hintergrund-Operationen (Auto-Save, Analytics) error toasten, aber UI nicht blockieren.

```
async function autoSave(draft) {
  try {
    await saveDraft(draft);
    // Kein Toast bei Erfolg (nicht stören)
  } catch (error) {
    toast.error('Entwurf konnte nicht gespeichert werden', {
      duration: 5000,
      action: {
        label: 'erneut',
        onClick: () => autoSave(draft),
      },
    });
  }
}
```

## Barrierefreiheit

Loading States müssen für alle Nutzer zugänglich sein.

### **prefers-reduced-motion**

**Wichtig:** Nutzer mit Bewegungsüberempfindlichkeit (Vestibular Disorders) sollten animationsfreie Versionen erhalten.

```
@media (prefers-reduced-motion: reduce) {
  .spinner {
    animation: none;
    opacity: 0.5; /* Statischer Spinner, halbtransparent */
  }

  @keyframes shimmer {
    /* Keine Animation */
  }

  progress:indeterminate::-webkit-progress-bar {
    animation: none;
    background: linear-gradient(90deg, var(--progress-track-background) 0%, var(--progress-bar-back
  }
}
```

### **aria-busy**

Container mit loadendem Content sollte `aria-busy="true"` haben:

```
<div aria-busy={loading}>
  {loading ? <Skeleton /> : <Content />}
</div>
```

Screenreader: Erkennt, dass dieser Bereich aktuell lädt.

## Live-Region Ankündigungen

Bei wichtigen Lade-Abschlüssen: Screenreader-Announcement via `aria-live`:

```
function ProductList() {
  const [products, setProducts] = useState([]);
  const [loading, setLoading] = useState(true);

  return (
    <>
      {/* Live-Region für Ankündigung */}
      <div role="status" aria-live="polite" className="sr-only">
        {loading ? 'Produkte werden geladen...' : `${products.length} Produkte geladen`}
      </div>

      <div aria-busy={loading}>
        {loading ? <ProductGridSkeleton /> : <ProductGrid products={products} />}
      </div>
    </>
  );
}
```

CSS für `.sr-only` (Screen Reader Only):

```
.sr-only {
  position: absolute;
  width: 1px;
  height: 1px;
  padding: 0;
  margin: -1px;
  overflow: hidden;
  clip: rect(0, 0, 0, 0);
  white-space: nowrap;
  border-width: 0;
}
```

## Code-Beispiele

### DelayedSpinner Component

Vollständige Spinner-Komponente mit Delay:

```

interface SpinnerProps {
  size?: 'sm' | 'md' | 'lg' | 'xl';
  color?: 'primary' | 'light' | 'on-primary';
  delay?: number;
  label?: string;
}

export function Spinner({ size = 'md', color = 'primary', delay = 200, label = 'Wird geladen' }: SpinnerProps) {
  const [show, setShow] = useState(delay === 0);

  useEffect(() => {
    if (delay === 0) return;

    const timer = setTimeout(() => setShow(true), delay);
    return () => clearTimeout(timer);
  }, [delay]);

  if (!show) return null;

  return (
    <div role="status" aria-live="polite" aria-label={label}>
      <svg
        className={`spinner spinner-${size} spinner-${color}`}
        width={`var(--spinner-size-${size})`}
        height={`var(--spinner-size-${size})`}
        viewBox="0 0 24 24"
        xmlns="http://www.w3.org/2000/svg"
      >
        <circle cx="12" cy="12" r="10" stroke="var(--spinner-track-color)" strokeWidth="var(--spinner-stroke-width)" fill="none" strokeDasharray="31.4 31.4" strokeLinecap="round" className="spinner-circle" />
      </svg>
    </div>
  );
}

```

CSS:

```
.spinner-circle {
  animation: spin var(--spinner-animation-duration) var(--spinner-animation-timing) infinite;
  transform-origin: center;
}

@keyframes spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

@media (prefers-reduced-motion: reduce) {
  .spinner-circle {
    animation: none;
    opacity: 0.5;
  }
}
```

## ProgressBar Component

```
interface ProgressBarProps {
  value?: number; // undefined = indeterminate
  max?: number;
  label?: string;
  showPercentage?: boolean;
}

export function ProgressBar({ value, max = 100, label, showPercentage = true }: ProgressBarProps) {
  const isDeterminate = value !== undefined;
  const percentage = isDeterminate ? Math.round((value / max) * 100) : undefined;
  const isComplete = percentage === 100;

  return (
    <div className="progress-container">
      {label && (
        <label className="progress-label-text">
          {label}
          {showPercentage && isDeterminate && ` (${percentage}%)`}
        </label>
      )}
      <progress
        className={isComplete ? 'progress-success' : ''}
        value={isDeterminate ? value : undefined}
        max={isDeterminate ? max : undefined}
        aria-valuenow={isDeterminate ? value : undefined}
        aria-valuemin={isDeterminate ? 0 : undefined}
        aria-valuemax={isDeterminate ? max : undefined}
        aria-label={label || 'Lädt...'}
      >
        {isDeterminate ? `${percentage}%` : 'Lädt...'}
      </progress>
    </div>
  );
}
```

## Optimistic Update Pattern

```

function useLike(postId: string, initialLikes: number, initialLiked: boolean) {
  const [optimisticState, setOptimistic] = useOptimistic(
    { likes: initialLikes, liked: initialLiked },
    (current, action: 'like' | 'unlike') => ({
      likes: current.likes + (action === 'like' ? 1 : -1),
      liked: action === 'like',
    })
  );

  const [isPending, startTransition] = useTransition();

  async function toggleLike() {
    const action = optimisticState.liked ? 'unlike' : 'like';

    startTransition(async () => {
      setOptimistic(action);

      try {
        await updateLike(postId, action === 'like');
      } catch (error) {
        toast.error('Like konnte nicht gespeichert werden', {
          action: {
            label: 'Erneut',
            onClick: () => toggleLike(),
          },
        });
      }
    });
  }

  return { ...optimisticState, toggleLike, isPending };
}

// Verwendung
function Post({ postId, initialLikes, initialLiked }) {
  const { likes, liked, toggleLike, isPending } = useLike(postId, initialLikes, initialLiked);

  return (
    <button onClick={toggleLike} disabled={isPending} className={liked ? 'liked' : ''}>
      ❤️ {likes}
    </button>
  );
}

```

## Anti-Patterns

Häufige Fehler bei Ladezuständen:

### 1. Spinner-Flash für schnelle Operationen

**Problem:** Spinner erscheint und verschwindet sofort (<200ms), erzeugt visuelles Flackern.

**Lösung:** spinner.delay (200ms) verwenden – Spinner erscheint erst nach Verzögerung.

## 2. Hardcoded Skeleton-Dimensionen

**Problem:** Skeleton hat feste Größen, finale Content-Größe weicht ab → Layout Shift.

**Lösung:** Verwende `react-loading-skeleton` (auto-sizing) oder matche Skeleton exakt auf finale Größe.

## 3. Optimistic UI für destruktive Aktionen

**Problem:** "Löschen" optimistisch zeigen – bei Fehler ist Item schon aus Liste verschwunden, Rollback verwirrend.

**Lösung:** Destruktive Aktionen NIEMALS optimistisch. Immer warten auf Server-Bestätigung.

## 4. Keine prefers-reduced-motion Unterstützung

**Problem:** Spinner und Shimmer-Animationen lösen bei manchen Nutzern Übelkeit aus.

**Lösung:** `@media (prefers-reduced-motion: reduce)` – Animationen deaktivieren oder stark reduzieren.

## 5. Indeterminate Progress für bekannte Dauer

**Problem:** Upload mit bekannter Dateigröße zeigt indeterminate Bar – Nutzer hat keine Info über Fortschritt.

**Lösung:** Determinate Progress mit Percentage verwenden, wenn Fortschritt berechenbar ist.

## 6. Zu viele gleichzeitige Spinner

**Problem:** Jeder kleine Bereich hat eigenen Spinner – UI wirkt chaotisch.

**Lösung:** Einen zentralen Spinner für Seiten-Load, lokale Spinner nur für isolierte Aktionen.

## 7. Fehlende Error Recovery

**Problem:** Laden schlägt fehl, Spinner verschwindet, Nutzer sieht leere Seite ohne Info.

**Lösung:** Error State mit klarer Message + Retry-Button anzeigen.

---

Phase: 05 - Feedback & System Responses Zuletzt aktualisiert: 2026-01-29 Version: 1.0

---

# Barrierefreiheit im Hydrophon Design-System

---

Barrierefreiheit (Accessibility, a11y) ist keine nachträgliche Ergänzung, sondern integraler Bestandteil jeder Komponente im Hydrophon Design-System.

**Commitment:** Alle Komponenten erfüllen WCAG 2.1 Level AA.

---

## Unser Ansatz

Das Hydrophon Design-System erfüllt WCAG 2.1 Level AA. Barrierefreiheit ist keine nachträgliche Ergänzung, sondern integraler Bestandteil jeder Komponente.

**Warum Barrierefreiheit im B2B-Bereich wichtig ist:**

- **Gesetzliche Anforderungen:** EU Web Accessibility Directive, Barrierefreiheitsstärkungsgesetz (BFSG)
  - **Breitere Nutzerbasis:** 15% der Weltbevölkerung leben mit Behinderungen
  - **Bessere UX für alle:** Klare Fokusindikatoren, Tastaturnavigation, verständliche Fehlermeldungen helfen allen Nutzern
  - **Baustellenkontext:** B2B-Sanitärbereich bedeutet oft rauе Umgebungen – große Touch-Targets, hohe Kontraste, Tastaturzugang sind essentiell
- 

## Grundprinzipien (POUR)

### 1. Wahrnehmbar (Perceivable)

Informationen und Benutzeroberflächen-Komponenten müssen in einer Weise präsentiert werden, die Nutzer wahrnehmen können.

**Unsere Umsetzung:**

- **Textalternativen:** Alle Bilder haben `alt`-Attribute, dekorative Icons haben `aria-hidden="true"`
- **Farbkontraste:** Mindestens 4.5:1 für normalen Text, 3:1 für große Texte und grafische Elemente
- **Nicht nur Farbe:** Informationen werden nie nur durch Farbe vermittelt (Farbe + Icon + Text)
  - Beispiel: Fehlermeldungen = Rote Farbe + AlertCircle-Icon + Text
  - Beispiel: Erfolg = Grüne Farbe + CheckCircle-Icon + Text

**Farbkontraste im Detail:**

Element	WCAG-Minimum	Hydrophon-Werte	Status
Normaler Text (< 18px)	4.5:1	Grau #575656 auf Weiß = 5.9:1	✓ Erfüllt
Großer Text (≥18px bold)	3:1	Grau #575656 auf Weiß = 5.9:1	✓ Erfüllt
Fokus-Indikatoren	3:1	Blau #008BD2 = 4.5:1	✓ Erfüllt
Fehlerfarbe	4.5:1	Rot auf Weiß = 4.8:1	✓ Erfüllt
Primary Button Text	4.5:1	Weiß auf Blau #008BD2 = 4.9:1	✓ Erfüllt

## 2. Bedienbar (Operable)

Benutzeroberflächen-Komponenten und Navigation müssen bedienbar sein.

### Unsere Umsetzung:

- **Tastaturzugriff:** Alle interaktiven Elemente sind per Tastatur erreichbar (Tab/Shift+Tab)
- **Fokus-Indikatoren:** Deutlich sichtbar mit 2px Outline, 2px Offset, 3:1 Kontrast
- **Keine Tastaturfallen:** ESC schliesst alle Overlays (Modals, Drawer, Tooltips)
- **Touch-Targets:** Mindestens 44×44px (WCAG AAA) für mobile und Baustellenumgebung
- **Keine Zeit-Limits:** Keine automatischen Timeouts ohne Warnung (außer Sitzungstimeouts mit Verlängerungsoption)

### Tasturnavigation im Detail:

Taste	Aktion	Beispiel
Tab	Nächstes fokussierbares Element	Durch Formular navigieren
Shift+Tab	Vorheriges fokussierbares Element	Zurück zum vorherigen Feld
Enter/Space	Aktiviert Button/Link	Button klicken
ESC	Schließt Modal/Drawer/Tooltip	Modal schließen
Pfeiltasten	Navigation in Radio-Groups	Radio-Option wählen

## 3. Verständlich (Understandable)

Informationen und Bedienung der Benutzeroberfläche müssen verständlich sein.

### Unsere Umsetzung:

- **Konsistente Navigation:** Header, Footer, Breadcrumb folgen denselben Patterns
- **Vorhersehbares Verhalten:** Gleiche Aktionen führen zu gleichen Ergebnissen
- **Hilfreiche Fehlermeldungen:** Deutsch, erklärend, mit Beispielen
  - Schlecht: "Ungültige Eingabe"
  - Gut: "Bitte gib eine gültige E-Mail-Adresse ein (z.B. [name@firma.de](mailto:name@firma.de))"
- **Labels und Anweisungen:** Jedes Formularfeld hat ein sichtbares Label
- **Fehlerprävention:** Bestätigungs-Dialoge für destruktive Aktionen

## 4. Robust (Robust)

Inhalte müssen robust genug sein, um von verschiedenen User Agents, einschließlich assistiver Technologien, zuverlässig interpretiert werden zu können.

### Unsere Umsetzung:

- **Valides HTML:** Semantische HTML-Elemente (button, nav, header, footer, main)
- **Korrekte ARIA-Attribute:** Nur wo nötig, native HTML bevorzugt
- **Kompatibilität:** Getestet mit NVDA (Windows), VoiceOver (macOS), JAWS (Windows)

## Kritische ARIA-Patterns

Diese Tabelle dokumentiert die wichtigsten ARIA-Implementierungen im Design-System:

Komponente	ARIA-Pattern	Details
Modal	<code>role="dialog"</code> , <code>aria-modal="true"</code>	Focus Trap aktiv, ESC zum Schließen, Fokus auf Close-Button bei Öffnung
Toast (Erfolg/Info)	<code>role="status"</code>	Polite announcement (unterbricht Screenreader nicht)
Toast (Warnung/Fehler)	<code>role="alert"</code>	Assertive announcement (unterbricht Screenreader sofort)
Breadcrumb	<code>&lt;nav aria-label="Breadcrumb"&gt;</code> , <code>aria-current="page"</code>	Markiert aktuelle Seite, eindeutiges Label
Mobile Drawer	<code>aria-expanded</code> , <code>aria-controls</code>	Toggle-Button zeigt Drawer-Status an
Formularfelder	<code>aria-describedby</code> , <code>aria-invalid</code>	Verknüpft Hilfetext/Fehler mit Eingabefeld
Header Navigation	<code>aria-current="page"</code>	Server-side gerendert für aktive Links
Checkbox/Radio	Native Input mit <code>opacity: 0</code>	Erhält Accessibility-Tree, NICHT <code>display: none</code> verwenden
Tooltip	Radix UI Pattern	Automatisch WCAG 1.4.13 konform (hoverable, dismissible, persistent)
Progress Bar	Native <code>&lt;progress&gt;</code>	Eingebaute Screenreader-Unterstützung

## Focus-Management

### Grundregeln

- Logische Tab-Reihenfolge:** Links nach rechts, oben nach unten (folgt F-Pattern)
- Fokus springt in Modal/Drawer:** Bei Öffnung erhält erstes Element (meist Close-Button) den Fokus
- Fokus kehrt zurück:** Bei Schließen springt Fokus zurück zum Trigger-Element
- Focus Trap in Modals:** Tab bleibt innerhalb des Modals, ESC zum Schließen

## Implementierung: Focus Trap

```
// Focus Trap Pattern (vereinfacht)
const modal = document.querySelector('[role="dialog"]');
const focusableElements = modal.querySelectorAll(
  'button, [href], input, select, textarea, [tabindex]:not([tabindex="-1"])'
);
const firstElement = focusableElements[0];
const lastElement = focusableElements[focusableElements.length - 1];

// Tab am Ende -> zurück zum Anfang
lastElement.addEventListener('keydown', (e) => {
  if (e.key === 'Tab' && !e.shiftKey) {
    e.preventDefault();
    firstElement.focus();
  }
});

// Shift+Tab am Anfang -> zum Ende
firstElement.addEventListener('keydown', (e) => {
  if (e.key === 'Tab' && e.shiftKey) {
    e.preventDefault();
    lastElement.focus();
  }
});

// ESC schließt Modal
modal.addEventListener('keydown', (e) => {
  if (e.key === 'Escape') {
    closeModal();
  }
});
```

**Empfehlung:** Verwende battle-tested Libraries wie Radix UI Dialog, die Focus-Traps automatisch implementieren.

## Fokus-Indikatoren

Alle interaktiven Elemente zeigen einen deutlich sichtbaren Fokus-Indikator.

### Standard-Fokus:

```
/* Alle interaktiven Elemente */
button:focus-visible,
a:focus-visible,
input:focus-visible,
select:focus-visible,
textarea:focus-visible {
  outline: 2px solid var(--color-focus); /* Hydrophon Blau #008BD2 */
  outline-offset: 2px;
}
```

### WCAG 2.2 Fokus-Anforderungen:

- **Kontrast:** Mindestens 3:1 gegen angrenzende Farben
- **Sichtbarkeit:** Mindestens 2px Breite
- **Offset:** 2px Abstand zum Element (verhindert Überlappung mit Content)

#### Hydrophon-Umsetzung:

- Fokus-Farbe: Hydrophon Blau #008BD2 (4.5:1 Kontrast)
  - Fokus-Breite: 2px
  - Fokus-Offset: 2px
  - Verwendung von `:focus-visible` (nur bei Tastatur, nicht bei Maus/Touch)
- 

## Tastatur-Navigation

### Globale Shortcuts

Taste	Aktion	Kontext
Tab	Nächstes fokussierbares Element	Global
Shift+Tab	Vorheriges fokussierbares Element	Global
Enter	Aktiviert Link oder Button	Buttons, Links
Space	Aktiviert Button, togglet Checkbox	Buttons, Checkboxes
ESC	Schließt Overlay	Modals, Drawer, Tooltips
Pfeiltasten	Navigation in Radio-Groups	Radio-Button-Gruppen

### Komponenten-spezifische Shortcuts

#### Modal-Dialog:

- Tab : Durch interaktive Elemente im Modal
- ESC : Modal schließen
- Focus Trap: Tab bleibt im Modal

#### Mobile Drawer:

- Tab : Durch Navigation-Links
- ESC : Drawer schließen
- Focus Trap: Tab bleibt im Drawer

#### Radio-Buttons:

- ↑/↓ oder ←/→ : Zwischen Radio-Optionen wechseln (und automatisch selektieren)
- Tab : Zur nächsten Fieldset-Gruppe

#### Tooltip:

- Hover oder Focus : Tooltip anzeigen
- ESC : Tooltip schließen
- Tooltip bleibt bei Hover über Tooltip-Inhalt sichtbar

## Screenreader-Unterstützung

### Getestete Screenreader

Plattform	Screenreader	Status
Windows	NVDA	✓ Getestet
Windows	JAWS	✓ Getestet
macOS	VoiceOver	✓ Getestet

### Screenreader-Patterns

Live Regions (Toast-Benachrichtigungen):

```
<!-- Erfolg/Info: Polite (unterbricht nicht) -->
<div role="status" aria-live="polite">
  Änderungen gespeichert
</div>

<!-- Warnung/Fehler: Assertive (unterbricht sofort) -->
<div role="alert" aria-live="assertive">
  Fehler beim Speichern
</div>
```

Labels und Beschreibungen:

```
<!-- Formularfeld mit Label und Hilfetext -->
<label for="email">
  E-Mail-Adresse
  <span class="required" aria-label="Pflichtfeld">*</span>
</label>
<input
  type="email"
  id="email"
  aria-describedby="email-help"
/>
<span id="email-help">Wir geben deine E-Mail nicht weiter.</span>

<!-- Fehlerfall -->
<input
  type="email"
  id="email"
  aria-invalid="true"
  aria-describedby="email-error"
/>
<span id="email-error" role="alert">
  Bitte gib eine gültige E-Mail-Adresse ein (z.B. name@firma.de)
</span>
```

## Motion & Animation

Alle Animationen respektieren die `prefers-reduced-motion` Einstellung.

CSS-Pattern:

```
/* Standard: Animation */
.modal {
  animation: fadeIn 200ms ease-out;
}

/* Reduced Motion: Keine Animation */
@media (prefers-reduced-motion: reduce) {
  .modal {
    animation: none;
  }
}

/* Spinner: Statisch statt rotierend */
@media (prefers-reduced-motion: reduce) {
  .spinner {
    animation: none;
    opacity: 0.6; /* Zeigt "Loading" visuell, aber statisch */
  }
}
```

Betroffene Komponenten:

- **Spinner:** Rotation wird ausgeschaltet, statisches Icon
- **Skeleton:** Shimmer-Animation wird ausgeschaltet
- **Modal:** Fade + Scale wird zu instant anzeigen
- **Toast:** Slide-in wird zu fade-in
- **Drawer:** Slide-out wird zu fade

## Weiterführende Dokumentation

Komponenten-spezifische Accessibility:

- [WCAG 2.1 AA Compliance pro Komponente](#) — Detaillierte WCAG-Kriterien für jede Komponente
- [Testing-Anleitung](#) — Praktische Checklisten für automatisierte und manuelle Tests

Externe Ressourcen:

- [WCAG 2.1 Richtlinien \(W3C\)](#)
- [ARIA Authoring Practices Guide \(APG\)](#)
- [WebAIM Contrast Checker](#)

Letzte Aktualisierung: 2026-01-29

# WCAG 2.1 AA Compliance

---

Komponenten-spezifische Accessibility-Anforderungen und deren Umsetzung im Hydrophon Design-System.

Diese Dokumentation zeigt, welche WCAG-Kriterien für jede Komponente relevant sind und wie das Design-System sie erfüllt.

---

## Compliance-Übersicht

Komponente	WCAG Level	Erfüllt	Details
Buttons	AA	✓	<a href="#">Details</a>
Text Inputs	AA	✓	<a href="#">Details</a>
Textarea	AA	✓	<a href="#">Details</a>
Select	AA	✓	<a href="#">Details</a>
Checkbox	AA	✓	<a href="#">Details</a>
Radio Button	AA	✓	<a href="#">Details</a>
Modal	AA	✓	<a href="#">Details</a>
Toast	AA	✓	<a href="#">Details</a>
Tooltip	AA	✓	<a href="#">Details</a>
Header Navigation	AA	✓	<a href="#">Details</a>
Mobile Drawer	AA	✓	<a href="#">Details</a>
Breadcrumb	AA	✓	<a href="#">Details</a>
Footer	AA	✓	<a href="#">Details</a>
Product Card	AA	✓	<a href="#">Details</a>
Content Card	AA	✓	<a href="#">Details</a>
Data Table	AA	✓	<a href="#">Details</a>
Loading States	AA	✓	<a href="#">Details</a>

---

## Buttons

### Relevante WCAG-Kriterien

- **1.4.3 Kontrast (Minimum):** 4.5:1 für Text, 3:1 für grafische Elemente
- **2.1.1 Tastatur:** Alle Buttons per Enter/Space aktivierbar
- **2.4.7 Fokus sichtbar:** Deutlich sichtbare Fokus-Indikatoren
- **2.5.5 Zielgröße (AAA):** Mindestens 44×44px Touch-Targets

## Umsetzung

### Kontrast:

- Primary Button: Weiß auf Blau (#008BD2) = 4.9:1 Kontrast ✓
- Secondary Button: Blau (#008BD2) auf Weiß = 4.5:1 Kontrast ✓
- Tertiary Button: Blau (#008BD2) auf Weiß = 4.5:1 Kontrast ✓
- Disabled State: neutral.400 auf neutral.200 = 2.1:1 (erlaubt für disabled)

### Tastatur:

- Native `<button>` Element: Enter und Space aktivieren Button automatisch
- `:focus-visible` für Fokus-Indikator (nur bei Tastatur)

### Fokus-Indikator:

```
.button:focus-visible {
  outline: 2px solid var(--button-focus-outline-color); /* Blau #008BD2 */
  outline-offset: 2px;
}
```

- Fokus-Ring: 2px Outline mit 2px Offset
- Kontrast: 4.5:1 gegen Weiß

### Touch-Targets:

- Medium Button (Default): 40px Höhe (WCAG AA erfüllt)
- Large Button: 48px Höhe (WCAG AAA erfüllt)
- Icon-only Buttons: 44px Mindestgröße (WCAG AAA erfüllt)

## Vollständige Spezifikation

→ [Button-Dokumentation](#)

---

## Text Inputs

### Relevante WCAG-Kriterien

- **1.3.1 Info und Beziehungen:** Label mit `for / id` verknüpft
- **1.4.1 Farbe nicht allein:** Fehler mit Farbe + Icon + Text kommuniziert
- **3.3.1 Fehlererkennung:** Fehler werden identifiziert und beschrieben
- **3.3.2 Labels oder Anweisungen:** Jedes Feld hat sichtbares Label

## Umsetzung

### Label-Verknüpfung:

```
<label for="email">E-Mail-Adresse</label>
<input type="email" id="email" />
```

- `<label for="field-id">` Verknüpfung für Screenreader

- Label immer sichtbar (kein Placeholder als Label)

#### Fehler-Kommunikation (dreifach):

- 1. Farbe:** Rote Border ( color.error )
- 2. Icon:** AlertCircle (Lucide, 16px) rechts im Input
- 3. Text:** Fehlertext unter dem Input

```
<input
  type="email"
  id="email"
  aria-invalid="true"
  aria-describedby="email-error"
  class="input--error"
/>
<span id="email-error" role="alert">
  Bitte gib eine gültige E-Mail-Adresse ein (z.B. name@firma.de)
</span>
```

#### Hilfetext-Verknüpfung:

```
<input
  type="email"
  id="email"
  aria-describedby="email-help"
/>
<span id="email-help">
  Wir geben deine E-Mail nicht weiter.
</span>
```

#### Pflichtfelder:

```
<label for="name">
  Name
  <span class="required" aria-label="Pflichtfeld">*</span>
</label>
<input
  type="text"
  id="name"
  required
  aria-required="true"
/>
```

## Vollständige Spezifikation

→ Text-Input Dokumentation → Validierung → Labels & Helper Text

## Textarea

### Relevante WCAG-Kriterien

- 1.3.1 Info und Beziehungen: Label verknüpft, Character Counter beschrieben
- 3.3.2 Labels oder Anweisungen: Zeichenlimit kommuniziert

### Umsetzung

Character Counter:

```
<label for="description">Beschreibung</label>
<textarea
  id="description"
  maxlength="500"
  aria-describedby="description-counter"
></textarea>
<span id="description-counter" aria-live="polite">
  0 / 500 Zeichen
</span>
```

- `aria-live="polite"` : Screenreader kündigt Zeichenzahl an (nicht-unterbrechend)
- Dynamische Farbcodierung: Grün (< 80%), Gelb (80-100%), Rot (100%)

### Vollständige Spezifikation

→ [Textarea-Dokumentation](#)

---

## Select

### Relevante WCAG-Kriterien

- 1.3.1 Info und Beziehungen: Label verknüpft
- 2.1.1 Tastatur: Native Tastaturnavigation (Pfeiltasten, Enter)

### Umsetzung

Native Select bevorzugt:

```
<label for="role">Rolle</label>
<select id="role">
  <option value="">Bitte wählen</option>
  <option value="admin">Administrator</option>
  <option value="user">Benutzer</option>
</select>
```

- Native `<select>` hat eingebaute Accessibility (Tastatur, Screenreader, Mobile)

- Custom Selects nur mit vollständiger ARIA-Implementierung (z.B. Radix UI Select)

#### Custom Select ARIA-Pattern:

- `role="combobox"` auf Trigger-Button
- `aria-expanded="true/false"` zeigt Status
- `aria-controls` verknüpft mit Listbox-ID
- `role="listbox"` auf Options-Container
- `role="option"` auf jeder Option

## Vollständige Spezifikation

→ [Select-Dokumentation](#)

---

## Checkbox

### Relevante WCAG-Kriterien

- 1.3.1 Info und Beziehungen: Fieldset/Legend für Gruppen
- 2.1.1 Tastatur: Space zum Toggeln
- 4.1.2 Name, Rolle, Wert: Native Input erhalten (nicht verbergen mit `display:none`)

### Umsetzung

#### Native Input (Anti-Pattern-Vermeidung):

```
/* Correct: Visually hidden, but accessible */
input[type="checkbox"] {
  opacity: 0;
  position: absolute;
}

/* WRONG: Breaks accessibility */
input[type="checkbox"] {
  display: none; /* ❌ Removes from accessibility tree */
}
```

- Native `<input type="checkbox">` mit `opacity: 0` (NICHT `display: none`)
- Erhält Accessibility-Tree und Keyboard-Navigation

#### Checkbox-Gruppen:

```
<fieldset>
  <legend>Interessen</legend>
  <label>
    <input type="checkbox" name="interests" value="sanitaer" />
    Sanitärinstallation
  </label>
  <label>
    <input type="checkbox" name="interests" value="heizung" />
    Heizung
  </label>
</fieldset>
```

- `<fieldset> + <legend>` für semantische Gruppierung
- Screenreader lesen Legend als Kontext

Fokus-Indikator:

```
input[type="checkbox"]:focus-visible + .checkbox-visual {
  outline: 2px solid var(--checkbox-focus-outline-color);
  outline-offset: 2px;
}
```

## Vollständige Spezifikation

→ [Checkbox-Dokumentation](#)

---

## Radio Button

### Relevante WCAG-Kriterien

- 1.3.1 Info und Beziehungen: Fieldset/Legend für Gruppen
- 2.1.1 Tastatur: Pfeiltasten zum Navigieren, Space zum Selektieren
- 4.1.2 Name, Rolle, Wert: Native Input erhalten

### Umsetzung

Native Input:

```
/* Correct: Visually hidden, but accessible */
input[type="radio"] {
  opacity: 0;
  position: absolute;
}
```

- Gleiche Regel wie Checkbox: `opacity: 0`, NICHT `display: none`

Radio-Gruppen:

```
<fieldset>
  <legend>Versandart</legend>
  <label>
    <input type="radio" name="shipping" value="standard" />
    Standard (3-5 Tage)
  </label>
  <label>
    <input type="radio" name="shipping" value="express" />
    Express (1-2 Tage)
  </label>
</fieldset>
```

#### Keyboard-Navigation:

- ↑/↓ oder ←/→ : Zwischen Radio-Optionen wechseln (und automatisch selektieren)
- Tab : Zur nächsten Radio-Gruppe (nicht zwischen Optionen)
- Native Browser-Verhalten bleibt erhalten

### Vollständige Spezifikation

→ [Radio Button-Dokumentation](#)

---

## Modal

#### Relevante WCAG-Kriterien

- 2.1.2 Keine Tastaturfalle: ESC zum Schließen
- 2.4.3 Fokusreihenfolge: Fokus springt in Modal bei Öffnung
- 4.1.2 Name, Rolle, Wert: `role="dialog"`, `aria-modal="true"`, `aria-labelledby`

#### Umsetzung

ARIA-Pattern:

```
<div
  role="dialog"
  aria-modal="true"
  aria-labelledby="modal-title"
>
  <h2 id="modal-title">Artikel löschen?</h2>
  <p>Dieser Vorgang kann nicht rückgängig gemacht werden.</p>
  <button>Abbrechen</button>
  <button>Löschen</button>
</div>
```

#### Focus-Management:

1. **Bei Öffnung:** Fokus springt auf erstes interaktives Element (meist Close-Button)
2. **Focus Trap:** Tab bleibt innerhalb des Modals (kein Zugriff auf Hintergrund)
3. **ESC-Taste:** Schließt Modal

4. Bei Schließen: Fokus kehrt zum Trigger-Element zurück

#### Radix UI Dialog (Empfohlen):

- Battle-tested Accessibility out-of-the-box
- Focus Trap automatisch
- ESC-Handling automatisch
- Portal-Rendering (außerhalb DOM-Hierarchie)

## Vollständige Spezifikation

→ [Modal-Dokumentation](#)

---

## Toast Notifications

### Relevante WCAG-Kriterien

- 4.1.3 Statusmeldungen: Screenreader-Ankündigung ohne Fokus-Änderung

### Umsetzung

#### ARIA Live Regions:

```
<!-- Erfolg/Info: Polite announcement -->
<div role="status" aria-live="polite">
  Änderungen gespeichert
</div>

<!-- Warnung/Fehler: Assertive announcement -->
<div role="alert" aria-live="assertive">
  Fehler beim Speichern
</div>
```

#### Severity-basiertes ARIA:

Toast-Typ	ARIA-Role	aria-live	Verhalten
Success	status	polite	Unterbricht Screenreader nicht
Info	status	polite	Unterbricht Screenreader nicht
Warning	alert	assertive	Unterbricht Screenreader sofort
Error	alert	assertive	Unterbricht Screenreader sofort

#### Fehler-Toasts bleiben sichtbar:

- Success/Info/Warning: Auto-dismiss nach 3-5 Sekunden
- Error: duration: 0 (bleibt sichtbar, manuelles Schließen erforderlich)
- Kritische Fehler erfordern Nutzer-Reaktion

#### Sonner-Library:

- ARIA Live Regions automatisch
- Pause-on-Hover (WCAG 1.4.13)

- Keyboard-DDismiss (ESC schließt alle Toasts)

## Vollständige Spezifikation

→ [Toast-Dokumentation](#)

---

## Tooltips

### Relevante WCAG-Kriterien

- 1.4.13 Inhalt bei Hover oder Fokus: Tooltip bleibt bei Hover sichtbar, ESC schließt

### Umsetzung

WCAG 1.4.13 Anforderungen:

1. **Dismissible**: ESC schließt Tooltip
2. **Hoverable**: Tooltip bleibt sichtbar bei Hover über Tooltip-Content
3. **Persistent**: Tooltip bleibt sichtbar bis Nutzer Hover/Fokus entfernt oder ESC drückt

Radix UI Tooltip (Empfohlen):

- WCAG 1.4.13 compliant out-of-the-box
- Keyboard-Zugriff: Fokus auf Trigger zeigt Tooltip
- Smart Positioning: Floating UI für Viewport-Collision-Detection
- Delay: 300ms (verhindert Flackern)

Fokus-Trigger:

```
<button aria-describedby="tooltip">
  Info
</button>
<div id="tooltip" role="tooltip">
  Hilfreicher Text
</div>
```

- Tooltip erscheint bei Hover UND Fokus
- `aria-describedby` verknüpft Trigger mit Tooltip-Text

## Vollständige Spezifikation

→ [Tooltip-Dokumentation](#)

---

## Header Navigation

### Relevante WCAG-Kriterien

- 2.4.4 Linkzweck (im Kontext): Aussagekräftige Link-Texte
- 4.1.2 Name, Rolle, Wert: `aria-current="page"` für aktive Links

## Umsetzung

Aktive Seite:

```
<nav>
  <a href="/" aria-current="page">Home</a>
  <a href="/produkte">Produkte</a>
  <a href="/kontakt">Kontakt</a>
</nav>
```

- `aria-current="page"` markiert aktuell aktive Seite
- Server-side gerendert (verhindert Flash of Incorrect State)
- Visuelle Markierung (Underline) + ARIA für Screenreader

Fokus-Indikatoren:

```
.header-nav a:focus-visible {
  outline: 2px solid var(--color-focus);
  outline-offset: 2px;
}
```

## Vollständige Spezifikation

→ [Header-Dokumentation](#)

---

## Mobile Drawer

### Relevante WCAG-Kriterien

- 2.1.2 Keine Tastaturfalle: ESC schließt Drawer
- 4.1.2 Name, Rolle, Wert: `aria-expanded`, `aria-controls` auf Toggle-Button

## Umsetzung

Toggle-Button:

```
<button
  aria-expanded="false"
  aria-controls="mobile-drawer"
  aria-label="Navigation öffnen"
>
  <svg><!-- Hamburger Icon --></svg>
</button>
```

- `aria-expanded="true/false"` : Zeigt Drawer-Status an
- `aria-controls` : Verknüpft Button mit Drawer-ID
- `aria-label` : Beschreibender Text für Screenreader

Drawer-Container:

```

<div
  id="mobile-drawer"
  role="dialog"
  aria-modal="true"
  aria-label="Hauptnavigation"
>
  <button aria-label="Schließen">
    <svg><!-- X Icon --></svg>
  </button>
  <nav>
    <!-- Navigation Links -->
  </nav>
</div>

```

#### Focus-Management:

- Fokus springt auf Close-Button bei Öffnung
- Focus Trap aktiv (Tab bleibt im Drawer)
- ESC schließt Drawer
- Fokus kehrt zu Hamburger-Button zurück bei Schließen

## Vollständige Spezifikation

→ Mobile Drawer-Dokumentation

---

## Breadcrumb

### Relevante WCAG-Kriterien

- 2.4.8 Position (Orientierung): Nutzer weiß wo er sich befindet
- 4.1.2 Name, Rolle, Wert: `aria-current="page"` für aktuelle Seite

### Umsetzung

ARIA-Pattern:

```

<nav aria-label="Breadcrumb">
  <ol>
    <li><a href="/">Home</a></li>
    <li><a href="/produkte">Produkte</a></li>
    <li aria-current="page">Armaturen</li>
  </ol>
</nav>

```

- `<nav aria-label="Breadcrumb">` : Eindeutiges Landmark-Label (unterscheidet von Header-Nav)
- `<ol>` : Semantische geordnete Liste
- `aria-current="page"` : Markiert aktuelle Seite (nicht als Link)

Separator:

```
li::after {
  content: ">";
  aria-hidden: true;
}
```

- CSS ::after für Separator (nicht im DOM)
- Screenreader ignoriert Separator

## Vollständige Spezifikation

→ [Breadcrumb-Dokumentation](#)

---

## Footer

### Relevante WCAG-Kriterien

- 2.4.1 Blöcke überspringen: Skip-Links ermöglichen schnelle Navigation
- 4.1.2 Name, Rolle, Wert: <footer> Landmark, mehrere <nav> mit eindeutigen Labels

### Umsetzung

Footer-Landmark:

```
<footer>
  <nav aria-label="Produkte">
    <!-- Produkt-Links -->
  </nav>
  <nav aria-label="Unternehmen">
    <!-- Unternehmens-Links -->
  </nav>
  <nav aria-label="Rechtliches">
    <!-- Legal-Links -->
  </nav>
</footer>
```

- <footer> : Automatisches role="contentinfo" Landmark
- Mehrere <nav> mit eindeutigen aria-label für Screenreader-Navigation

Touch-Target-Optimierung:

```
.footer-nav a {
  line-height: 2; /* 28px bei 14px Font = WCAG AA */
}
```

## Vollständige Spezifikation

→ [Footer-Dokumentation](#)

## Product Card

### Relevante WCAG-Kriterien

- 1.1.1 Nicht-Text-Inhalt: Produktbilder haben `alt`-Attribute
- 2.4.4 Linkzweck: Gesamte Card klickbar mit aussagekräftigem Link-Text

### Umsetzung

Alt-Texte:

```

```

- Beschreibender `alt`-Text (Produktnname + Variante)
- `loading="lazy"` : Performance-Optimierung
- `decoding="async"` : Non-blocking Image-Decode

Klickbare Card:

```
<article>
  <a href="/produkte/armatur-xy" class="card-link">
    
    <h3>Einhebel-Waschtischarmatur XY</h3>
    <p>ab 129,00 €</p>
  </a>
</article>
```

- Gesamte Card als `<a>` (nicht Button in Card)
- Heading als Link-Text für Screenreader-Kontext

### Vollständige Spezifikation

→ [Product Card-Dokumentation](#)

---

## Content Card

### Relevante WCAG-Kriterien

- 1.1.1 Nicht-Text-Inhalt: Bilder haben `alt`-Attribute
- 2.4.4 Linkzweck: Aussagekräftige Call-to-Action

## Umsetzung

Bilder (optional):

```
<!-- Mit Bild -->


<!-- Ohne Bild (Text-only Variante) -->
<!-- Kein img-Element, kein alt nötig -->
```

Call-to-Action:

```
<a href="/blog/case-study">
  Mehr erfahren →
</a>
```

- Aussagekräftiger Link-Text (nicht nur "Hier klicken")
- Pfeil-Icon rein dekorativ ( `aria-hidden` im Icon-SVG)

## Vollständige Spezifikation

→ Content Card-Dokumentation

---

## Data Table

### Relevante WCAG-Kriterien

- 1.3.1 Info und Beziehungen: Header-Zellen mit `scope="col"`-Attribut
- 2.1.1 Tastatur: Horizontaler Scroll für responsive Tabellen

## Umsetzung

Table-Struktur:

```
<table>
  <thead>
    <tr>
      <th scope="col">Produkt</th>
      <th scope="col">Preis</th>
      <th scope="col">Verfügbarkeit</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">Armatur XY</th>
      <td>129,00 €</td>
      <td>Auf Lager</td>
    </tr>
  </tbody>
</table>
```

- `scope="col"` auf `<th>` in `<thead>`
- `scope="row"` auf erste Spalte (Produkt-Namen)
- Screenreader lesen Kontext: "Armatur XY, Preis: 129,00 €"

**Responsive: Horizontal Scroll:**

```
<div class="table-wrapper" tabindex="0">
  <table>
    <!-- Table content -->
  </table>
</div>
```

- `tabindex="0"` : Macht Scroll-Container keyboard-accessible
- Alternative: Stacked-Card-Layout für sehr einfache Tabellen

## Vollständige Spezifikation

→ [Data Table-Dokumentation](#)

---

## Loading States

### Relevante WCAG-Kriterien

- 2.2.2 Pausieren, Stoppen, Ausblenden: Animationen respektieren `prefers-reduced-motion`
- 4.1.3 Statusmeldungen: Progress-Updates werden angekündigt

### Umsetzung

Spinner:

```

.spinner {
  animation: rotate 1s linear infinite;
}

@media (prefers-reduced-motion: reduce) {
  .spinner {
    animation: none;
    opacity: 0.6; /* Statisches Icon */
  }
}

```

#### Progress Bar (Native):

```

<progress
  value="60"
  max="100"
  aria-label="Upload-Fortschritt"
>
  60%
</progress>

```

- Native `<progress>` Element hat eingebaute Screenreader-Unterstützung
- `aria-label` : Beschreibt was geladen wird

#### Skeleton Screens:

- `prefers-reduced-motion: reduce` : Shimmer-Animation deaktiviert
- Statische Grau-Boxen zeigen Loading-Zustand

### Vollständige Spezifikation

→ [Loading-Dokumentation](#)

---

### Zusammenfassung

Alle 17 dokumentierten Komponenten erfüllen WCAG 2.1 Level AA.

#### Häufigste Accessibility-Patterns:

1. Native HTML bevorzugen: `<button>` , `<select>` , `<progress>` , `<fieldset>` haben eingebaute Accessibility
2. ARIA nur wo nötig: Korrekte `role` , `aria-expanded` , `aria-current` , `aria-describedby`
3. Focus-Management: Deutliche Fokus-Indikatoren, Focus Traps in Modals, Fokus-Rückkehr
4. Keyboard-Support: Tab, Enter, Space, ESC, Arrow-Keys
5. Screenreader-Support: Live Regions, Labels, Beschreibungen
6. Motion-Respekt: `prefers-reduced-motion` für alle Animationen

#### Testing-Verantwortung:

Jede neue Komponente oder Feature muss vor Release:

- ✓ axe DevTools: 0 kritische Fehler
- ✓ Lighthouse Accessibility: Score 90+
- ✓ Keyboard-Test: Alle Funktionen erreichbar

- ✓ Screenreader-Test: Inhalte verständlich vorgelesen

→ [Testing-Anleitung](#)

---

**Letzte Aktualisierung:** 2026-01-29

---

# Accessibility Testing Guide

---

Praktische Anleitung zur Überprüfung der WCAG 2.1 AA Compliance im Hydrophon Design-System.

**Ziel:** Sicherstellen, dass alle Komponenten für alle Nutzer zugänglich sind — unabhängig von Sehvermögen, Mobilität oder verwendeter Technologie.

---

## Test-Strategie

Accessibility-Testing folgt der **30/70-Regel**:

- **30%** der Probleme werden durch automatisierte Tools gefunden
- **70%** erfordern manuelle Prüfung

**Warum beide nötig sind:**

- **Automatisierte Tools** finden technische Fehler (fehlende `alt`-Attribute, Kontrast-Probleme, ungültige ARIA)
- **Manuelle Tests** prüfen Nutzbarkeit (ist die Tab-Reihenfolge logisch? sind Labels verständlich? funktioniert Keyboard-Navigation?)

**Empfohlener Workflow:**

1. **Automatisiert testen** (axe DevTools + Lighthouse) — findet schnell offensichtliche Fehler
  2. **Keyboard-Test** — prüft ob alle Funktionen per Tastatur erreichbar sind
  3. **Screenreader-Test** — prüft ob Inhalte verständlich vorgelesen werden
  4. **Kontrast-Prüfung** — prüft Farbkontraste manuell
  5. **Zoom-Test** — prüft Nutzbarkeit bei 200% Zoom
  6. **Motion-Test** — prüft `prefers-reduced-motion` Respekt
- 

## Automatisierte Tests

### axe DevTools (Empfohlen)

Browser-Extension für Chrome, Firefox, Edge.

**Installation:**

1. Chrome Web Store öffnen
2. "axe DevTools" suchen
3. Extension installieren
4. DevTools öffnen (F12)
5. Tab "axe DevTools" wählen

**Verwendung:**

1. Seite oder Komponente laden
2. "Analyze" klicken
3. Ergebnisse nach Schweregrad sortieren:
  - **Critical:** Sofort beheben (blockiert Nutzer)

- **Serious:** Priorität (beeinträchtigt Nutzbarkeit)
- **Moderate:** Wichtig (kann verwirren)
- **Minor:** Nice-to-have (verbessert UX)

4. Issues beheben und erneut testen

#### Was axe findet:

✓ Fehlende `alt`-Texte auf Bildern ✓ Kontrastprobleme (Text zu hell) ✓ Fehlende Labels auf Formularfeldern ✓ Ungültige ARIA-Attribute ✓ Fehlende Landmarks (`<nav>`, `<main>`, `<footer>`) ✓ Fokusreihenfolge-Probleme (`tabindex` Chaos) ✓ Redundante ARIA (z.B. `role="button"` auf `<button>`)

#### Was axe NICHT findet:

✗ Ob Labels **sinnvoll** sind (nur dass sie existieren) ✗ Ob Fokusreihenfolge **logisch** ist (nur dass sie existiert) ✗ Ob Inhalte **verständlich** sind ✗ Komplexe Interaktionsmuster (Modals, Drawers, Custom Widgets) ✗ Keyboard-Traps

**Ziel:** 0 kritische und serious Fehler vor Release.

---

## Lighthouse

In Chrome DevTools integriert.

#### Verwendung:

1. Chrome DevTools öffnen (F12)
2. Tab "Lighthouse" wählen
3. "Categories" → nur "Accessibility" auswählen
4. "Analyze page load" klicken
5. Report durchlesen

#### Lighthouse-Score:

- **90-100:** Excellent (Hydrophon-Ziel)
- **50-89:** Needs improvement
- **0-49:** Poor

Lighthouse gibt konkrete Handlungsempfehlungen mit Links zu WCAG-Dokumentation.

#### Unterschied zu axe:

- axe: Detaillierter, mehr Tests, bessere Erklärungen
- Lighthouse: Schneller Überblick, Performance + A11y zusammen

**Empfehlung:** Beide verwenden (axe für Details, Lighthouse für Score-Tracking).

---

## Manuelle Tests

### 1. Tastatur-Navigation

#### Warum wichtig:

- 15% der Nutzer verwenden keine Maus (Motorik, Präferenz, Power-User)
- Screenreader-Nutzer navigieren primär per Tastatur
- B2B-Kontext: Baustellenumgebung, Handschuhe, schmutzige Hände

#### Checkliste:

- Alle interaktiven Elemente per Tab erreichbar
- Fokus-Indikator immer sichtbar (kein `outline: none` ohne Alternative)

- Logische Tab-Reihenfolge (links→rechts, oben→unten)
- Enter/Space aktiviert Buttons und Links
- ESC schließt Modals/Drawers/Tooltips
- Pfeiltasten funktionieren in Radio-Groups
- Keine Tastaturfallen (Fokus kann immer weiterbewegt werden)
- Skip-Links vorhanden (direkt zu Hauptinhalt springen)

**Test-Methode:**

1. Maus weglassen / Touch deaktivieren
2. Mit Tab durch Seite navigieren
3. Alle Funktionen nur per Tastatur ausführen:
  - Formular ausfüllen
  - Modal öffnen und schließen
  - Navigation verwenden
  - Buttons klicken
  - Tooltips öffnen

**Häufige Probleme:**

Problem	Ursache	Lösung
Fokus nicht sichtbar	outline: none im CSS	:focus-visible mit 2px Outline
Falsche Tab-Reihenfolge	tabindex > 0 verwendet	tabindex entfernen, DOM-Reihenfolge anpassen
Element nicht fokussierbar	<div> statt <button>	Natives HTML-Element verwenden
Tastatur-Falle in Modal	Kein Focus Trap	Radix UI Dialog verwenden

## 2. Screenreader-Test

**Warum wichtig:**

- 2% der Nutzer verwenden Screenreader (Blindheit, Sehbehinderung)
- Prüft ob ARIA-Attribute korrekt sind
- Prüft ob Inhalte verständlich sind

**Empfohlene Screenreader:**

Plattform	Screenreader	Kosten	Marktanteil
Windows	NVDA	Kostenlos	72%
Windows	JAWS	~2000€/Jahr	53%
macOS	VoiceOver	Integriert (Cmd+F5)	11%

**NVDA Installation (Windows):**

1. <https://www.nvaccess.org/> besuchen
2. "Download" klicken
3. Installer ausführen
4. NVDA starten (Ctrl+Alt+N)

**Grundlegende NVDA-Befehle:**

Taste	Aktion
Caps Lock + Leertaste	NVDA-Modus wechseln (Browse/Focus)
Tab	Nächstes fokussierbares Element
H	Nächste Überschrift (Shift+H für vorherige)
B	Nächster Button
E	Nächstes Eingabefeld
K	Nächster Link
Caps Lock + ↓	Zeile vorlesen
Caps Lock + T	Titel vorlesen

**VoiceOver (macOS):**

- **Aktivieren:** Cmd+F5
- **Navigation:** Ctrl+Option+→ (nächstes Element)
- **Aktivieren:** Ctrl+Option+Space
- **Rotor:** Ctrl+Option+U (Liste aller Überschriften, Links, etc.)

**Test-Checkliste:**

- Seitenstruktur wird korrekt vorgelesen (Überschriften-Hierarchie)
- Landmarks werden identifiziert ( `<nav>` , `<main>` , `<footer>` )
- Bilder haben sinnvolle `alt`-Texte (nicht nur "Bild" oder Dateiname)
- Formularfelder werden mit Label vorgelesen
- Fehler werden angekündigt ( `role="alert"` )
- Statusänderungen werden angekündigt (Toast-Notifications mit `role="status"` )
- Modal-Inhalte werden vorgelesen ( `role="dialog"` , `aria-labelledby` )
- Aktuelle Seite wird markiert ( `aria-current="page"` )
- Buttons haben aussagekräftige Namen (nicht nur Icon)

**Häufige Probleme:**

Problem	Ursache	Lösung
"Button" ohne Name	Icon-Button ohne <code>aria-label</code>	<code>aria-label="Schließen"</code> hinzufügen
Fehler nicht angekündigt	Fehler ohne <code>role="alert"</code>	<code>role="alert"</code> auf Fehlermeldung
Formular-Chaos	Labels nicht verknüpft	<code>&lt;label for="id"&gt;</code> verwenden
"Bild" vorgelesen	<code>alt=""</code> oder Dateiname	Beschreibenden alt -Text schreiben

### 3. Kontrastprüfung

**Warum wichtig:**

- 8% der Männer haben Farbsehschwäche (Rot-Grün-Blindheit)
- Ältere Nutzer haben reduziertes Sehvermögen
- B2B-Kontext: Sonnenlicht auf Baustelle, schlechte Bildschirme

**Tools:**

- WebAIM Contrast Checker: <https://webaim.org/resources/contrastchecker/>

- Vorder- und Hintergrundfarbe eingeben
- Sofort WCAG AA/AAA Status sehen
- **Chrome DevTools:**
  - Element inspizieren
  - "Computed" → "Contrast ratio" zeigt automatisch AA/AAA Status

#### WCAG-Mindestanforderungen:

Element	WCAG AA	WCAG AAA	Hydrophon
Normaler Text (< 18px)	4.5:1	7:1	5.9:1 ✓
Großer Text (≥18px bold oder ≥24px)	3:1	4.5:1	5.9:1 ✓
Grafische Elemente (Icons, Fokus)	3:1	—	4.5:1 ✓

#### Hydrophon-Farbkontraste:

Kombination	Kontrast	Status
Grau #575656 auf Weiß	5.9:1	✓ AA (Normal)
Weiß auf Blau #008BD2	4.9:1	✓ AA (Normal)
Fokus-Ring Blau #008BD2 auf Weiß	4.5:1	✓ AA (Grafik)
Fehler-Rot auf Weiß	4.8:1	✓ AA (Normal)

#### Test-Methode:

1. Screenshot der Komponente machen
2. Farben mit Eyedropper-Tool (Chrome DevTools) identifizieren
3. In WebAIM Contrast Checker eingeben
4. Prüfen ob Minimum erfüllt ist

#### Häufige Probleme:

- Hellgraue Placeholder-Texte (oft < 4.5:1)
- Disabled States (erlaubt < 4.5:1)
- Sekundäre Texte zu hell
- Links ohne Underline UND zu wenig Kontrast

## 4. Zoom-Test

#### Warum wichtig:

- Sehbehinderungen erfordern 200% Zoom
- WCAG 2.1 fordert Nutzbarkeit bis 200% Zoom
- Ältere Nutzer zoomen häufig

#### WCAG-Anforderung:

*Content can be zoomed up to 200% without loss of content or functionality, and without requiring horizontal scrolling.*

#### Test-Methode:

1. Browser auf 200% zoomen (Cmd/Ctrl + + +)
2. Prüfen:

- Kein Text abgeschnitten?
- Kein horizontales Scrollen nötig?
- Alle Funktionen erreichbar?
- Layout responsiv (kein fixed-width Chaos)?

### 3. Funktionen testen (Modal öffnen, Formular absenden, etc.)

#### Häufige Probleme:

- Fixed-width Container schneiden Text ab
- Buttons überlappen
- Fixed Header verdeckt Inhalt
- Horizontal Scroll erscheint

#### Hydrophon-Lösung:

- Responsive Grid mit `max-width` statt `width`
  - Relative Units (`rem`, `em`) statt `px` für Text
  - Flexible Layouts mit Flexbox/Grid
- 

## 5. Motion-Test

#### Warum wichtig:

- Vestibulare Störungen: Animationen lösen Schwindel, Übelkeit aus
- WCAG 2.1 (2.3.3): Motion from Interactions (Level AAA, but recommended)
- Einige Nutzer aktivieren `prefers-reduced-motion` dauerhaft

#### Test-Methode:

1. In Betriebssystem "Reduzierte Bewegung" aktivieren:
  - **macOS:** Systemeinstellungen → Bedienungshilfen → Anzeige → Bewegung reduzieren
  - **Windows:** Einstellungen → Erleichterte Bedienung → Anzeige → Animationen deaktivieren
2. Seite neu laden (wichtig!)
3. Prüfen:
  - Animationen reduziert oder entfernt?
  - Spinner statisch (keine Rotation)?
  - Skeleton ohne Shimmer?
  - Modal fade-in instant?
  - Toast slide-in wird zu fade-in?
4. Prüfen: Funktionalität erhalten? (Loading-Zustände erkennbar?)

#### Hydrophon-Pattern:

```
.spinner {
  animation: rotate 1s linear infinite;
}

@media (prefers-reduced-motion: reduce) {
  .spinner {
    animation: none;
    opacity: 0.6; /* Statisches Icon, aber sichtbar */
  }
}
```

**Betroffene Komponenten:**

- Spinner (Rotation → statisch)
  - Skeleton (Shimmer → statisch)
  - Modal (Scale+Fade → instant)
  - Toast (Slide → Fade)
  - Drawer (Slide → Fade)
  - Transitions (Reduziert auf 50ms oder entfernt)
- 

**Test-Checkliste für neue Komponenten**

Vor Release einer neuen Komponente diese Checkliste durchgehen:

**Automatisiert**

- axe DevTools: 0 kritische Fehler, 0 serious Fehler
- Lighthouse Accessibility: Score 90+ (Hydrophon-Ziel: 95+)

**Manuell — Tastatur**

- Alle interaktiven Elemente per Tab erreichbar
- Fokus-Indikator immer sichtbar (2px Outline, 3:1 Kontrast)
- Tab-Reihenfolge logisch (links→rechts, oben→unten)
- Enter/Space aktiviert Buttons
- ESC schließt Overlays (Modal, Drawer, Tooltip)
- Keine Tastaturfallen

**Manuell — Screenreader**

- NVDA oder VoiceOver Test durchgeführt
- Inhalte verständlich vorgelesen
- Labels vorhanden und sinnvoll
- Fehler werden angekündigt
- Status-Updates werden angekündigt (Toasts)
- ARIA-Attribute korrekt (role, aria-label, aria-describedby)

**Manuell — Visuell**

- Kontrast: Alle Texte  $\geq$  4.5:1 (normale),  $\geq$  3:1 (große)
- Fokus-Indikator:  $\geq$  3:1 Kontrast gegen Hintergrund
- Informationen nicht nur durch Farbe (Farbe + Icon + Text)
- Zoom: 200% funktional, kein horizontaler Scroll
- Reduced Motion: Animationen respektieren Einstellung

**Komponenten-spezifisch****Modal/Drawer:**

- Focus Trap aktiv (Tab bleibt im Modal)
- ESC schließt Modal

- Fokus kehrt zu Trigger zurück
- `role="dialog"`, `aria-modal="true"`

**Formularfelder:**

- Label mit `for` / `id` verknüpft
- Fehler mit `aria-invalid` und `aria-describedby`
- Pflichtfelder mit `required` und visueller Markierung (\*)

**Toasts:**

- `role="status"` (Erfolg/Info) oder `role="alert"` (Warnung/Fehler)
- Fehler-Toasts bleiben sichtbar (kein Auto-dismiss)

## Häufige Fehler und Lösungen

### Formular-Fehler

Fehler	Beispiel	Lösung
"Form elements must have labels"	<input> ohne <label>	<label for="id"> hinzufügen
"Form fields must have error messages"	Fehler nur visuell (rot)	<code>aria-describedby="error-id" + &lt;span id="error-id" role="alert"&gt;</code>
"Placeholder is not a label"	Placeholder als einziger Text	Sichtbares <label> hinzufügen

### Interaktive Elemente

Fehler	Beispiel	Lösung
"Interactive controls must be focusable"	<div onclick="...">	<button> verwenden
"Button has no accessible name"	Icon-Button ohne Text	<code>aria-label="Schließen"</code>
"Links must have discernible text"	<a href="#">Mehr</a> ohne Kontext	Kontext im Link: "Mehr über Produkt XY"

### ARIA-Fehler

Fehler	Beispiel	Lösung
"ARIA attributes must have valid values"	<code>aria-expanded="yes"</code>	<code>aria-expanded="true"</code> (Boolean)
"ARIA role not allowed"	<button role="button">	<code>role</code> entfernen (redundant)
"aria-describedby ID does not exist"	Tippfehler in ID	IDs prüfen und synchronisieren

## Kontrast-Fehler

Fehler	Beispiel	Lösung
"Elements must have sufficient color contrast"	Hellgrau #AAAAAA auf Weiß = 2.3:1	Dunkleres Grau #757575 (4.5:1)
"Focus indicator must be visible"	outline: none ohne Alternative	:focus-visible { outline: 2px solid blue; }

## Struktur-Fehler

Fehler	Beispiel	Lösung
"Page must have one main landmark"	Kein <main>	<main> um Hauptinhalt
"Heading levels should not skip"	<h1> → <h3>	<h2> einfügen
"Landmarks must have unique labels"	Zwei <nav> ohne Label	aria-label="Hauptnavigation" + aria-label="Footer-Navigation"

## Testing-Tools Übersicht

Tool	Typ	Kosten	Verwendung
axe DevTools	Browser Extension	Kostenlos	Automatisierte Tests, detaillierte Fehlerberichte
Lighthouse	Chrome DevTools	Kostenlos	Schneller Score, Performance + A11y
NVDA	Screenreader (Windows)	Kostenlos	Manuelle Screenreader-Tests
JAWS	Screenreader (Windows)	~2000€/Jahr	Manuelle Tests (Marktführer)
VoiceOver	Screenreader (macOS)	Integriert	Manuelle Tests (macOS/iOS)
WebAIM Contrast Checker	Online Tool	Kostenlos	Kontrast-Prüfung
WAVE	Browser Extension	Kostenlos	Visuelles Overlay, zeigt Fehler inline

## Weiterführende Ressourcen

### Offizielle Dokumentation:

- [WCAG 2.1 Quick Reference](#) — Alle WCAG-Kriterien mit Erklärungen
- [ARIA Authoring Practices Guide](#) — ARIA-Patterns für Widgets
- [WebAIM Articles](#) — Praktische Anleitungen

### Testing-Tools:

- [axe DevTools](#) — Browser Extension
- [Lighthouse](#) — Chrome DevTools
- [NVDA Screenreader](#) — Kostenloser Screenreader

### Hydrophon-spezifisch:

- [Accessibility Overview](#) — Grundprinzipien und ARIA-Patterns
  - [WCAG Compliance](#) — Komponenten-spezifische Compliance
- 

Letzte Aktualisierung: 2026-01-29

---

# Design Tokens

---

Zentrale Quelle für alle Gestaltungswerte im Hydrophon Design-System.

## Was sind Design Tokens?

Design Tokens sind benannte Variablen für Gestaltungswerte wie Farben, Abstände und Schriftgrößen. Sie ermöglichen konsistente Gestaltung und einfache Anpassungen.

Beispiel:

```
// Token-Definition
{
  "hydrophon": {
    "blau": {
      "500": {
        "$value": "#008BD2",
        "$type": "color"
      }
    }
  }
}
```

```
/* CSS-Variable (generiert) */
--hydrophon-blau-500: #008bd2;
```

## Token-Architektur

### Drei Ebenen



#### Warum drei Ebenen?

- **Primitives:** Reine Werte ohne Kontext (Farbskala 50-900)
- **Semantics:** Bedeutung im System (action, error, surface)
- **Components:** Konkrete Anwendung (button, input, modal)

Diese Architektur ermöglicht Flexibilität: Änderungen an Primitive Tokens propagieren automatisch durch das gesamte System, während Semantic und Component Tokens die Bedeutung klar kommunizieren.

## Token-Dateien

Datei	Inhalt	Tokens
colors.json	Farbpalette, Produktlinien, Semantische Farben	~80
typography.json	Schriftfamilien, Größen, Gewichte	~25
spacing.json	Abstände, Grid, Breakpoints	~30
effects.json	Schatten, Rundungen	~15
buttons.json	Button-Varianten und -Zustände	~50
forms.json	Formular-Komponenten	~50
navigation.json	Header, Footer, Breadcrumb	~25
feedback.json	Modal, Toast, Tooltip, Loading	~40

Gesamt: ~315 Design Tokens

Alle Token-Dateien befinden sich in `design-system/tokens/`.

## Token-Format

Wir verwenden das W3C Design Tokens Community Group (DTCG) Format:

```
{
  "token-name": {
    "$value": "wert",
    "$type": "color|dimension|fontFamily|..."
  }
}
```

### Unterstützte Typen:

- `color` - Farbwerte (#hex, rgb, hsl)
- `dimension` - Größen (px, rem)
- `fontFamily` - Schriftfamilien
- `fontWeight` - Schriftgewichte
- `number` - Zahlen (z.B. line-height)
- `duration` - Zeitwerte (ms, s)
- `shadow` - Schatteneffekte

## Referenzen (Aliase)

Tokens können andere Tokens referenzieren:

```
{
  "button": {
    "primary": {
      "background": {
        "default": {
          "$value": "{hydrophon.blau.500}",
          "$type": "color"
        }
      }
    }
  }
}
```

Die `{token.pfad}` Syntax erstellt Verknüpfungen. Ändert sich `hydrophon.blau.500`, aktualisieren sich alle Referenzen automatisch.

Beispiel aus generierten CSS:

```
--button-primary-background-default: var(--color-action-primary);
--color-action-primary: var(--hydrophon-blau-500);
--hydrophon-blau-500: #008bd2;
```

Diese Kette ermöglicht Theme-Anpassungen auf jeder Ebene.

## Build-Prozess

### Style Dictionary

Token-Kompilierung mit Style Dictionary:

```
# Installation (einmalig)
npm install

# Tokens kompilieren
npm run build
```

Ausgabe:

```
design-system/build/
├── css/
|   └── variables.css      # CSS Custom Properties
└── json/
    └── tokens.json        # Flaches JSON für Tools
```

## Konfiguration

`config.json`:

```
{
  "source": ["tokens/**/*.json"],
  "platforms": {
    "css": {
      "transformGroup": "css",
      "buildPath": "build/css/",
      "files": [
        {
          "destination": "variables.css",
          "format": "css/variables",
          "options": {
            "outputReferences": true
          }
        }
      ]
    }
  }
}
```

**outputReferences: true** bedeutet, dass CSS-Variablen Referenzen erhalten:

```
--button-primary-background-default: var(--hydrophon-blau-500);
```

Statt direkt eingebettete Werte:

```
--button-primary-background-default: #008bd2;
```

Dies erhält die Token-Beziehungen und ermöglicht einfachere Theme-Anpassungen.

## Verwendung

### CSS importieren

```
@import 'design-system/build/css/variables.css';

.meine-komponente {
  background-color: var(--color-background-primary);
  padding: var(--spacing-4);
  border-radius: var(--border-radius-md);
}
```

### HTML (Link-Tag)

```
<link rel="stylesheet" href="design-system/build/css/variables.css">
```

## JavaScript/TypeScript

```
// JSON-Tokens für Runtime-Zugriff
import tokens from 'design-system/build/json/tokens.json';

const primaryColor = tokens['hydrophon-blau-500'];
```

## Best Practices

### Immer Semantic Tokens verwenden

```
/* ✅ Gut - Semantische Bedeutung klar */
.button {
  background: var(--color-action-primary);
}

/* ❌ Schlecht - Primitive Token direkt */
.button {
  background: var(--hydrophon-blau-500);
}
```

### Spacing-Skala nutzen

```
/* ✅ Gut - Token aus Spacing-System */
.card {
  padding: var(--spacing-6);
  gap: var(--spacing-4);
}

/* ❌ Schlecht - Hartcodierte Werte */
.card {
  padding: 24px;
  gap: 16px;
}
```

## Responsive Anpassungen

```
/* Mobile */
.section {
  padding: var(--spacing-12) var(--spacing-4);
}

/* Desktop */
@media (min-width: 1024px) {
  .section {
    padding: var(--spacing-24) var(--spacing-8);
  }
}
```

## Weiterführende Dokumentation

- [CSS-Variablen Referenz](#) - Vollständige Variable-Liste
  - [Grid & Breakpoints](#) - Responsive Layout
-

# CSS-Variablen Referenz

---

Vollständige Liste aller generierten CSS Custom Properties.

## Import

```
@import 'design-system/build/css/variables.css';
```

Oder Link-Tag:

```
<link rel="stylesheet" href="design-system/build/css/variables.css">
```

## Farben

### Markenfarben

Hydrophon Blau:

Variable	Wert	Verwendung
--hydrophon-blau-50	#e5f5fd	Hintergrund, sehr hell
--hydrophon-blau-100	#b8e4f9	Hover-States, Borders
--hydrophon-blau-200	#8ad3f5	Deaktivierte Zustände
--hydrophon-blau-300	#5cc2f1	Focus-Indikatoren
--hydrophon-blau-400	#2eb1ed	Leichte Varianten
--hydrophon-blau-500	#008bd2	Primärfarbe (Brand CI)
--hydrophon-blau-600	#007bb8	Hover auf Primärfarbe
--hydrophon-blau-700	#006a9e	Active-States
--hydrophon-blau-800	#005a84	Sehr dunkel
--hydrophon-blau-900	#00496a	Text auf hellem Hintergrund

Hydrophon Grau:

Variable	Wert	Verwendung
--hydrophon-grau-50	#f7f7f7	Hintergrund, sehr hell
--hydrophon-grau-100	#e8e8e8	Hover-States
--hydrophon-grau-200	#d1d1d1	Borders
--hydrophon-grau-300	#b0afaf	Deaktiviert
--hydrophon-grau-400	#939292	Sekundäre UI-Elemente
--hydrophon-grau-500	#575656	<b>Sekundärfarbe (Brand CI)</b>
--hydrophon-grau-600	#4d4c4c	Hover
--hydrophon-grau-700	#434242	Active
--hydrophon-grau-800	#393838	Sehr dunkel
--hydrophon-grau-900	#2f2e2e	Text

## Neutralgrau-Skala

Variable	Wert	Verwendung
--neutral-50	#fafafa	Hintergrund, sehr hell
--neutral-100	#f5f5f5	Sekundärer Hintergrund
--neutral-200	#e5e5e5	Borders, Dividers
--neutral-300	#d4d4d4	Input-Borders
--neutral-400	#a3a3a3	Placeholder-Text
--neutral-500	#737373	Muted Text
--neutral-600	#525252	Sekundärer Text
--neutral-700	#404040	Body-Text
--neutral-800	#262626	Heading-Text
--neutral-900	#171717	Primärer Text (höchster Kontrast)
--neutral-white	#ffffff	Weiß
--neutral-black	#000000	Schwarz

## Semantische Farben

Variable	Wert	Verwendung
--color-primary	var(--hydrophon-blau-500)	Primäre Markenelemente
--color-secondary	var(--hydrophon-grau-500)	Sekundäre Markenelemente
--color-success	#22c55e	Erfolgsmeldungen, Bestätigungen
--color-warning	#f59e0b	Warnungen, wichtige Hinweise
--color-error	#ef4444	Fehler, destruktive Aktionen
--color-info	#3b82f6	Informationen, neutrale Hinweise

Hintergrundfarben:

Variable	Wert	Verwendung
--color-background-primary	var(--neutral-white)	Haupthintergrund
--color-background-secondary	var(--neutral-50)	Sekundärer Hintergrund
--color-background-tertiary	var(--neutral-100)	Tertiärer Hintergrund

Textfarben:

Variable	Wert	Verwendung
--color-text-primary	var(--neutral-900)	Haupttext (höchster Kontrast)
--color-text-secondary	var(--neutral-700)	Sekundärer Text
--color-text-muted	var(--neutral-500)	Gedämpfter Text
--color-text-inverse	var(--neutral-white)	Text auf dunklem Hintergrund

Aktionsfarben:

Variable	Wert	Verwendung
--color-action-primary	var(--hydrophon-blau-500)	Primäre Aktionen
--color-action-primary-hover	var(--hydrophon-blau-600)	Hover-State
--color-action-primary-active	var(--hydrophon-blau-700)	Active-State
--color-action-disabled	var(--neutral-200)	Deaktiviert
--color-text-on-primary	var(--neutral-white)	Text auf Primärfarbe
--color-text-disabled	var(--neutral-400)	Deaktivierter Text

## Produktlinien-Farben

Variable	Wert	Produktlinie
--productline-hyhero-primary	#f49a36	hyHero Orange
--productline-hyhero-secondary	#373643	hyHero Dunkelgrau
--productline-hyindustry-primary	#0e2638	hyIndustry Dunkelblau
--productline-gluy-primary	#ffeeb6	Gluy Gelb
--productline-gluy-secondary	#88a9c3	Gluy Hellblau
--productline-gluy-tertiary	#14202e	Gluy Dunkelblau

## Typografie

### Schriftfamilien

Variable	Wert
--font-family-sans	Inter, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif
--font-family-mono	'JetBrains Mono', 'Fira Code', 'Courier New', monospace

## Schriftgrößen

Variable	Wert	Verwendung
--font-size-xs	12px	Captions, Legal Text
--font-size-sm	14px	Labels, Form Inputs, UI Text
--font-size-base	16px	Body Text, Paragraphs
--font-size-lg	18px	Emphasis Body Text
--font-size-xl	20px	H5 Subheadings
--font-size-2xl	24px	H4 Headings
--font-size-3xl	30px	H3 Headings
--font-size-4xl	36px	H2 Headings
--font-size-5xl	48px	H1 Headings
--font-size-6xl	60px	Hero Headings

## Schriftgewichte

Variable	Wert	Verwendung
--font-weight-regular	400	Body Text
--font-weight-medium	500	UI-Elemente, Betonung
--font-weight-semibold	600	Subheadings
--font-weight-bold	700	Headings

## Zeilenhöhen

Variable	Wert	Verwendung
--line-height-tight	1.25	Große Headings
--line-height-snug	1.375	Subheadings
--line-height-normal	1.5	Body Text
--line-height-relaxed	1.625	Komfortables Lesen
--line-height-loose	2	Spezielle Layouts (Footer-Links)

## Buchstabenabstand

Variable	Wert	Verwendung
--letter-spacing-tighter	-0.05em	Display Text
--letter-spacing-tight	-0.025em	Headings
--letter-spacing-normal	0em	Body Text
--letter-spacing-wide	0.025em	UI-Elemente
--letter-spacing-wider	0.05em	Labels, Buttons, Uppercase

## Abstände

4px Basis-Einheit:

Variable	Wert	Berechnung	Verwendung
--spacing-0	0px	$0 \times 4\text{px}$	Kein Abstand
--spacing-0-5	2px	$0.5 \times 4\text{px}$	Extra tiny
--spacing-1	4px	$1 \times 4\text{px}$	Kleinstes Spacing
--spacing-1-5	6px	$1.5 \times 4\text{px}$	Tiny
--spacing-2	8px	$2 \times 4\text{px}$	Extra small
--spacing-2-5	10px	$2.5 \times 4\text{px}$	Small-medium
--spacing-3	12px	$3 \times 4\text{px}$	Small
--spacing-4	16px	$4 \times 4\text{px}$	Medium (häufig für Padding)
--spacing-5	20px	$5 \times 4\text{px}$	Medium-large
--spacing-6	24px	$6 \times 4\text{px}$	Large
--spacing-7	28px	$7 \times 4\text{px}$	Extra large
--spacing-8	32px	$8 \times 4\text{px}$	2XL
--spacing-9	36px	$9 \times 4\text{px}$	2.5XL
--spacing-10	40px	$10 \times 4\text{px}$	3XL
--spacing-11	44px	$11 \times 4\text{px}$	3.5XL (WCAG Touch Target)
--spacing-12	48px	$12 \times 4\text{px}$	4XL
--spacing-14	56px	$14 \times 4\text{px}$	5XL
--spacing-16	64px	$16 \times 4\text{px}$	6XL
--spacing-20	80px	$20 \times 4\text{px}$	7XL
--spacing-24	96px	$24 \times 4\text{px}$	8XL
--spacing-28	112px	$28 \times 4\text{px}$	9XL
--spacing-32	128px	$32 \times 4\text{px}$	10XL

## Effekte

### Border Radius

Variable	Wert	Verwendung
--border-radius-none	0px	Keine Rundung
--border-radius-sm	2px	Kleine Elemente
--border-radius-base	4px	Standard (Buttons, Inputs)
--border-radius-md	6px	Cards
--border-radius-lg	8px	Größere Cards, Modals
--border-radius-xl	12px	Pills, große Elemente
--border-radius-2xl	16px	Feature Cards
--border-radius-3xl	24px	Dekorative Elemente
--border-radius-full	9999px	Kreise, Pills

### Schatten

Variable	Wert	Verwendung
--shadow-none	none	Kein Schatten
--shadow-sm	0 1px 2px rgba(0,0,0,0.05)	Minimale Tiefe
--shadow-base	0 1px 3px rgba(0,0,0,0.1)	Standard-Elevation
--shadow-md	0 4px 6px rgba(0,0,0,0.1)	Elevated Elements
--shadow-lg	0 10px 15px rgba(0,0,0,0.1)	Floating Elements
--shadow-xl	0 20px 25px rgba(0,0,0,0.1)	Modals, Dialogs
--shadow-2xl	0 25px 50px rgba(0,0,0,0.25)	Dramatische Elevation
--shadow-inner	inset 0 2px 4px rgba(0,0,0,0.05)	Inset-Elemente

## Grid & Breakpoints

### Grid-Variablen

Variable	Wert	Verwendung
--grid-columns	12	Standard 12-Spalten-Grid
--grid-max-width	1280px	Maximale Content-Breite
--grid-margins-mobile	16px	Mobile Seitenränder
--grid-margins-tablet	32px	Tablet Seitenränder
--grid-margins-desktop	64px	Desktop Seitenränder
--grid-gutters-mobile	16px	Mobile Spaltenabstand
--grid-gutters-tablet	24px	Tablet Spaltenabstand
--grid-gutters-desktop	32px	Desktop Spaltenabstand

### Breakpoints

Variable	Wert	Geräte
--breakpoints-sm	640px	Landscape Phones
--breakpoints-md	768px	Tablets Portrait
--breakpoints-lg	1024px	Tablets Landscape, Laptops
--breakpoints-xl	1280px	Large Desktops
--breakpoints-2xl	1536px	Very Large Desktops

### Container

Variable	Wert
--container-sm	640px
--container-md	768px
--container-lg	1024px
--container-xl	1280px

## Komponenten-Tokens

### Buttons

Primary Button:

Variable	Wert
--button-primary-background-default	var(--color-action-primary)
--button-primary-background-hover	var(--color-action-primary-hover)
--button-primary-background-active	var(--color-action-primary-active)
--button-primary-background-disabled	var(--color-action-disabled)
--button-primary-foreground-default	var(--color-text-on-primary)
--button-primary-foreground-disabled	var(--color-text-disabled)

Größen:

Variable	Wert
--button-size-small-min-height	32px
--button-size-small-font-size	14px
--button-size-small-padding-x	var(--spacing-3)
--button-size-small-padding-y	6px
--button-size-medium-min-height	40px
--button-size-medium-font-size	16px
--button-size-large-min-height	48px
--button-size-large-font-size	18px

Base:

Variable	Wert
--button-base-border-radius	var(--border-radius-base)
--button-base-border-width	2px
--button-base-font-weight	500
--button-base-transition	150ms ease-in-out
--button-focus-outline-width	2px
--button-focus-outline-offset	2px

## Inputs

Variable	Wert
--input-background-default	var(--neutral-white)
--input-field-border-default	var(--neutral-300)
--input-field-border-focus	var(--hydrophon-blau-500)
--input-field-border-error	var(--color-error)
--input-height-sm	32px
--input-height-md	40px
--input-height-lg	48px
--input-focus-ring-width	3px
--input-base-border-radius	var(--border-radius-base)

## Icons

Variable	Wert	Verwendung
--icon-size-xs	16px	Inline mit kleinem Text
--icon-size-sm	20px	Buttons mit Text
--icon-size-md	24px	Standalone Icons
--icon-size-lg	32px	Hero Sections
--icon-size-xl	48px	Dekorativ
--icon-stroke-weight	2px	Strichstärke
--icon-touch-target-minimum	44px	WCAG AAA Touch Target

## Navigation

Variable	Wert
--navigation-header-height-desktop	80px
--navigation-header-height-mobile	64px
--navigation-drawer-width	280px
--navigation-drawer-z-index	1000
--navigation-toggle-size	44px

## Cards

Variable	Wert
--card-padding	var(--spacing-6)
--card-radius	var(--border-radius-lg)
--card-gap	var(--spacing-6)
--card-shadow-default	var(--shadow-sm)
--card-shadow-hover	var(--shadow-md)

## Modal

Variable	Wert
--modal-overlay-background	rgba(0, 0, 0, 0.5)
--modal-size-sm-width	400px
--modal-size-md-width	600px
--modal-size-lg-width	900px
--modal-z-index	1000
--modal-animation-duration	200ms

## Tooltip

Variable	Wert
--tooltip-background	var(--neutral-900)
--tooltip-color	var(--neutral-white)
--tooltip-font-size	var(--font-size-sm)
--tooltip-delay	300ms
--tooltip-z-index	1100

## Toast

Variable	Wert
--toast-width	360px
--toast-z-index	1200
--toast-duration-success	3000ms
--toast-duration-info	4000ms
--toast-duration-warning	5000ms
--toast-duration-error	0ms

## Loading

Variable	Wert
--spinner-size-sm	16px
--spinner-size-md	24px
--spinner-size-lg	32px
--spinner-size-xl	48px
--spinner-delay	200ms
--progress-height	8px

## Vollständige Liste

Generierte Datei: [design-system/build/css/variables.css](#)

Diese Dokumentation zeigt die wichtigsten CSS-Variablen nach Kategorie. Die vollständige Liste aller ~315 CSS-Variablen mit allen Zustandsvarianten befindet sich in der generierten Datei.

Verwendung:

```
.meine-komponente {
  /* Farben */
  background: var(--color-background-primary);
  color: var(--color-text-primary);

  /* Abstände */
  padding: var(--spacing-4);
  gap: var(--spacing-3);

  /* Typografie */
  font-family: var(--font-family-sans);
  font-size: var(--font-size-base);

  /* Effekte */
  border-radius: var(--border-radius-md);
  box-shadow: var(--shadow-sm);
}
```

## Weiterführende Dokumentation

- [Design Tokens](#) - Token-Architektur und Build-Prozess
- [Grid & Breakpoints](#) - Responsive Layout

# Grid & Breakpoints

---

Responsive Layout-System für konsistente Gestaltung.

## Grid-System

12-Spalten-Grid für maximale Flexibilität.

### Spaltenaufteilung

Spalten	Verwendung	Beispiel
12	Full Width	Hero-Sections, Artikel
6+6	Zweispaltig	Bild + Text, Features
4+4+4	Dreispaltig	Feature-Grid, Produkte
3+3+3+3	Vierspaltig	Icon-Grid, kleine Cards
8+4 oder 4+8	Asymmetrisch	Hauptinhalt + Sidebar

Das 12-Spalten-System ermöglicht alle gängigen Aufteilungen: 2, 3, 4, 6, 12 Spalten.

### CSS Grid Implementierung

```
.container {
  display: grid;
  grid-template-columns: repeat(12, 1fr);
  gap: var(--grid-gutters-desktop);
  max-width: var(--grid-max-width);
  margin: 0 auto;
  padding: 0 var(--grid-margins-desktop);
}

/* Spannende Spalten */
.col-12 { grid-column: span 12; } /* Full Width */
.col-6 { grid-column: span 6; } /* Half Width */
.col-4 { grid-column: span 4; } /* Third Width */
.col-3 { grid-column: span 3; } /* Quarter Width */
```

## Grid-Variablen

Variable	Mobile	Tablet	Desktop
--grid-columns	12	12	12
--grid-gutter	16px	24px	32px
--grid-margin	16px	32px	64px
--grid-max-width	-	-	1280px

**Gutter:** Abstand zwischen Spalten **Margin:** Seitenrand außen am Container **Max-Width:** Maximale Breite des Content-Bereichs

## Breakpoints

### Definitionen

Name	Variable	Wert	Geräte
Small	--breakpoints-sm	640px	Smartphones Landscape
Medium	--breakpoints-md	768px	Tablets Portrait
Large	--breakpoints-lg	1024px	Tablets Landscape, Laptops
Extra Large	--breakpoints-xl	1280px	Desktop Monitors
2XL	--breakpoints-2xl	1536px	Große Monitore

## Mobile-First Ansatz

Media Queries von klein nach groß:

```

/* Mobile (Standard) */
.element {
  font-size: 14px;
  padding: var(--spacing-4);
}

/* Tablet und größer (768px+) */
@media (min-width: 768px) {
  .element {
    font-size: 16px;
    padding: var(--spacing-6);
  }
}

/* Desktop und größer (1024px+) */
@media (min-width: 1024px) {
  .element {
    font-size: 18px;
    padding: var(--spacing-8);
  }
}

```

**Vorteil Mobile-First:** Kleinere Geräte laden weniger CSS, progressive Enhancement für größere Screens.

## Breakpoint-Variablen in CSS

```
/* Breakpoints als Custom Properties */
:root {
  --breakpoints-sm: 640px;
  --breakpoints-md: 768px;
  --breakpoints-lg: 1024px;
  --breakpoints-xl: 1280px;
  --breakpoints-2xl: 1536px;
}

/* HINWEIS: CSS Media Queries können keine var() nutzen */
/* Breakpoint-Werte direkt in Media Queries verwenden */
@media (min-width: 768px) { /* entspricht --breakpoints-md */ }
```

CSS Custom Properties funktionieren nicht in @media -Regeln, daher müssen Werte direkt verwendet werden.

## Responsive Patterns

### Container

```
.container {
  width: 100%;
  max-width: var(--grid-max-width);
  margin: 0 auto;
  padding-left: var(--grid-margins-mobile);
  padding-right: var(--grid-margins-mobile);
}

/* Tablet */
@media (min-width: 768px) {
  .container {
    padding-left: var(--grid-margins-tablet);
    padding-right: var(--grid-margins-tablet);
  }
}

/* Desktop */
@media (min-width: 1024px) {
  .container {
    padding-left: var(--grid-margins-desktop);
    padding-right: var(--grid-margins-desktop);
  }
}
```

### Responsive Grid mit auto-fit

Automatische Spaltenanzahl basierend auf Viewport:

```
.card-grid {
  display: grid;
  grid-template-columns: repeat(
    auto-fit,
    minmax(min(280px, 100%), 1fr)
  );
  gap: var(--spacing-6);
}
```

**Erklärung:**

- repeat(auto-fit, ...) - Automatische Spaltenanzahl
- minmax(min(280px, 100%), 1fr) - Minimum 280px, aber nie größer als Container
- min(280px, 100%) - Verhindert Overflow auf kleinen Screens

Keine Media Queries nötig - Grid passt sich automatisch an!

**Responsive Typography**

```
/* Mobile */
h1 {
  font-size: var(--font-size-3xl); /* 30px */
  line-height: var(--line-height-tight);
}

/* Desktop */
@media (min-width: 1024px) {
  h1 {
    font-size: var(--font-size-5xl); /* 48px */
  }
}
```

**Touch Targets**

WCAG 2.1 AA erfordert 44x44px Touch Targets:

Variable	Wert	Verwendung
--icon-touch-target-minimum	44px	WCAG AAA Empfehlung
--icon-touch-target-compact	32px	Nur Desktop mit ausreichend Abstand

```

/* Mobile - WCAG-konforme Touch Targets */
.button {
  min-height: 44px;
  min-width: 44px;
}

/* Desktop - kompaktere UI möglich */
@media (min-width: 1024px) {
  .button-small {
    min-height: 32px;
    min-width: 32px;
  }
}

```

## Spacing-Anpassungen

### Responsive Spacing

Spacing	Mobile	Desktop	Variable
Section Padding	48px	96px	spacing-12 / spacing-24
Component Gap	16px	24px	spacing-4 / spacing-6
Card Padding	16px	24px	spacing-4 / spacing-6
Grid Gutter	16px	32px	grid-gutters-*

### Beispiel

```

.section {
  /* Mobile */
  padding: var(--spacing-12) 0;
}

@media (min-width: 1024px) {
  .section {
    /* Desktop */
    padding: var(--spacing-24) 0;
  }
}

```

## Grid Margins & Gutters

```
.grid {  
  display: grid;  
  gap: var(--grid-gutters-mobile);  
  padding: 0 var(--grid-margins-mobile);  
}  
  
@media (min-width: 768px) {  
  .grid {  
    gap: var(--grid-gutters-tablet);  
    padding: 0 var(--grid-margins-tablet);  
  }  
}  
  
@media (min-width: 1024px) {  
  .grid {  
    gap: var(--grid-gutters-desktop);  
    padding: 0 var(--grid-margins-desktop);  
  }  
}
```

## Layout-Beispiele

### Hero Section

```
.hero {  
  display: grid;  
  grid-template-columns: repeat(12, 1fr);  
  gap: var(--grid-gutters-mobile);  
  padding: var(--spacing-12) var(--grid-margins-mobile);  
}  
  
.hero__content {  
  grid-column: span 12; /* Mobile: Full Width */  
}  
  
@media (min-width: 768px) {  
  .hero {  
    gap: var(--grid-gutters-tablet);  
    padding: var(--spacing-16) var(--grid-margins-tablet);  
  }  
  
  .hero__content {  
    grid-column: span 8; /* Tablet: 8 von 12 Spalten */  
  }  
}  
  
@media (min-width: 1024px) {  
  .hero {  
    gap: var(--grid-gutters-desktop);  
    padding: var(--spacing-24) var(--grid-margins-desktop);  
  }  
  
  .hero__content {  
    grid-column: span 6; /* Desktop: 6 von 12 Spalten */  
  }  
}
```

## Feature Grid

```
.features {  
  display: grid;  
  gap: var(--spacing-6);  
}  
  
/* Mobile: 1 Spalte */  
.features {  
  grid-template-columns: 1fr;  
}  
  
/* Tablet: 2 Spalten */  
@media (min-width: 768px) {  
  .features {  
    grid-template-columns: repeat(2, 1fr);  
  }  
}  
  
/* Desktop: 3 Spalten */  
@media (min-width: 1024px) {  
  .features {  
    grid-template-columns: repeat(3, 1fr);  
    gap: var(--spacing-8);  
  }  
}
```

## Sidebar Layout

```
.layout {
  display: grid;
  gap: var(--spacing-6);
}

/* Mobile: Gestapelt */
.layout__main,
.layout__sidebar {
  grid-column: span 12;
}

/* Desktop: Sidebar rechts */
@media (min-width: 1024px) {
  .layout {
    grid-template-columns: repeat(12, 1fr);
  }

  .layout__main {
    grid-column: span 8; /* 8 Spalten für Hauptinhalt */
  }

  .layout__sidebar {
    grid-column: span 4; /* 4 Spalten für Sidebar */
  }
}
```

## Weiterführende Dokumentation

- [Design Tokens](#) - Token-Architektur
  - [CSS-Variablen](#) - Vollständige Referenz
-

# Für Marketing-Teams

---

Schneller Zugang zu allem, was Sie für Ihre tägliche Arbeit brauchen.

## Einstieg

### Wenn Sie neu sind

**Empfohlen:** Starten Sie mit dem [Marketing Onboarding Guide](#) - Ein umfassender Leitfaden mit Logo-Verwendung, Farbsystem, Typografie und Do's and Don'ts.

## Schnellzugriff

Was Sie suchen	Wo Sie es finden
Logo-Regeln	<a href="#">Logo-Guidelines</a>
Markenfarben	<a href="#">Farbsystem</a>
Schriften	<a href="#">Typografie</a>
Logo-Dateien	<a href="#">design-system/assets/logo/hydrophon/svg/</a>

## Häufige Aufgaben

### Präsentation erstellen

- Schriftgrößen: [Typografie-Übersicht](#)
- Farben: [Primärfarben verwenden](#)
- Abstände: [Spacing-System](#)

### Social Media Post gestalten

- Mindestgrößen: [Logo Digital](#)
- Markenfarben: [Farbsystem](#)
- Icon-System: [Icons verwenden](#)

### Druckmaterial vorbereiten

- CMYK-Werte: [Farben für Print](#)
- Logo-Dateien: [PDF-Varianten](#)
- Schriftarten: [Typografie-System](#)

### Marketing-Website gestalten

- Buttons: [Button-System](#)
- Produktkarten: [Product Cards](#)

- Content Cards: Content Cards für Artikel/Blog

## Markenidentität verstehen

### Farbsystem

- Primärfarben (Hydrophon Blau, Gluy Grün)
- Sekundärfarben
- Funktionale Farben

### Typografie

- Schriftfamilie Inter
- Größenskala
- Semantische Textstile

### Logo-Guidelines

- Verwendung & Varianten
- Schutzraum
- Mindestgrößen
- Dateiformate

## Do's and Don'ts auf einen Blick

### Logo

→ Logo-Fehlverwendung vermeiden

### Farben

→ Farb-Fehler vermeiden

## Komponenten für Marketing-Material

### Interaktive Elemente

- Buttons (CTA, Links)

### Inhaltsdarstellung

- Product Cards - Produktpräsentation
- Content Cards - Artikel, News, Blog-Posts
- Data Tables - Produktvergleiche, Spezifikationen

## Brauchen Sie Hilfe?

Bei Fragen zur korrekten Markenverwendung wenden Sie sich an Ihr Design-Team oder konsultieren Sie die umfassende Dokumentation in den jeweiligen Komponenten-Bereichen.

**Zielgruppe:** Marketing-Teams, Content-Ersteller, Externe Agenturen **Zweck:** Schneller Zugang zu Markenressourcen für tägliche Marketing-Aufgaben

---

# Für Designer

---

Übersicht aller Komponenten, Tokens und Gestaltungsregeln.

## Design-Grundlagen

Thema	Dokumentation
Farbsystem	<a href="#">Farben &amp; Abstufungen</a>
Typografie	<a href="#">Schriftsystem</a>
Spacing & Grid	<a href="#">Abstands- und Rastersystem</a>
Effekte	<a href="#">Schatten &amp; Rundungen</a>
Icons	<a href="#">Icon-System</a>
Logo	<a href="#">Logo-Guidelines</a>

## Komponenten

### Interaktive Elemente

- [Buttons](#) - Primary, Secondary, Tertiary Varianten

### Formulare

- [Text-Inputs](#)
- [Textarea](#)
- [Select](#)
- [Checkbox](#)
- [Radio Buttons](#)
- [Labels & Hilfetexte](#)
- [Validierung](#)

→ [Formular-Übersicht](#)

### Navigation

- [Header](#)
- [Mobile Drawer](#)
- [Breadcrumb](#)
- [Footer](#)

→ [Navigation-Übersicht](#)

### Inhaltsdarstellung

- [Product Cards](#)

- Content Cards
- Data Tables

## Feedback & System

- Modal Dialoge
  - Tooltips
  - Toast Notifications
  - Loading States
- Feedback-Übersicht

## Wann welche Variante?

### Buttons

Aktion	Variante	Beispiel
Hauptaktion (1x pro Bereich)	Primary	"Jetzt kaufen", "Anfrage senden"
Alternative Aktionen	Secondary	"Mehr erfahren", "Zur Vergleichsliste"
Tertiäre/Abbrechen	Tertiary	"Abbrechen", "Zurück"

Faustregel: Ein Primary Button pro Viewport-Bereich. Mehrere Aktionen? → 1x Primary + Rest Secondary/Tertiary.

### Karten

Inhalt	Komponente	Wann verwenden
Produkt mit Bild	Product Card	E-Commerce, Produktlisten, Kataloge
Artikel/Blog	Content Card	News, Ratgeber, Blog-Posts
Vergleichsdaten	Data Table	Technische Spezifikationen, Produktvergleiche

### Formulare

Feldtyp	Komponente	Wann verwenden
Kurze Texteingabe	Text-Input	Name, E-Mail, Telefon
Mehrzeilige Eingabe	Textarea	Nachricht, Kommentar, Beschreibung
Auswahl aus Liste	Select	Land, Kategorie, Produktlinie
Mehrfachauswahl	Checkbox	Newsletter, Zustimmungen, Features
Einfachauswahl	Radio Button	Anrede, Zahlungsart, Versandoption

## Navigation

Kontext	Komponente	Wann verwenden
Hauptnavigation	Header	Top-Level Menü (Desktop)
Mobile Navigation	Mobile Drawer	Hamburger-Menü (Mobile/Tablet)
Seitenhierarchie	Breadcrumb	Tiefe Content-Strukturen (3+ Ebenen)
Seiten-Fußzeile	Footer	Links, Rechtliches, Kontakt

## Feedback

Situation	Komponente	Wann verwenden
Wichtige Entscheidung	Modal Dialog	Bestätigung, kritische Warnung, mehrstufige Formulare
Kurze Erklärung	Tooltip	Icon-Erklärung, Label-Ergänzung (1-5 Wörter)
System-Feedback	Toast	Erfolg/Fehler nach Aktion, nicht-blockierend
Ladevorgang	Loading States	Daten laden, Formular absenden, Seite wechseln

## Barrierefreiheit

### Umfassende Accessibility-Dokumentation:

- [Accessibility-Übersicht](#) - Prinzipien, WCAG-Grundlagen, Design-Checkliste
- [WCAG-Compliance](#) - Komponenten-spezifische Compliance-Details
- [Testing-Guide](#) - axe DevTools, manuelles Testing, Screen Reader

## Wichtige Prinzipien

### WCAG 2.1 AA Compliance:

- Alle interaktiven Elemente haben 44x44px Touch-Targets (WCAG 2.5.5)
- Farbkontrast mindestens 4.5:1 für Text, 3:1 für UI-Komponenten
- Fokus-Indikatoren: 2px Outline + 2px Offset für Tastaturnavigation
- Screen Reader-freundlich: Semantisches HTML, ARIA-Attribute

### In jeder Komponenten-Dokumentation:

- Accessibility-Sektion mit ARIA-Patterns
- Keyboard Navigation-Spezifikation
- Screen Reader-Verhalten

## Token-Referenz

### Design Tokens verstehen

Alle visuellen Eigenschaften sind als Design Tokens definiert:

colors.json	→ Farbpalette (50-900 Abstufungen)
typography.json	→ Schriften, Größen, Zeilenhöhen
spacing.json	→ Abstände (4px-Basis)
buttons.json	→ Button-spezifische Tokens
forms.json	→ Formular-spezifische Tokens
navigation.json	→ Navigation-spezifische Tokens
feedback.json	→ Feedback-spezifische Tokens

**Verwendung in Design-Tools:** → Tokens können in Figma/Sketch als Variablen importiert werden (zukünftige Integration)

## Markenidentität

### Logo-System

→ [Logo-Guidelines](#) - Alle Varianten, Schutzraum, Mindestgrößen

### Farbidentität

→ [Farbsystem](#) - Primär/Sekundär/Funktionale Farben mit Verwendungsrichtlinien

### Typografie

→ [Typografie](#) - Inter als Markenschrift, semantische Textstile

---

**Zielgruppe:** UI/UX Designer, Visuelle Gestalter **Zweck:** Schneller Zugriff auf alle Komponenten mit "Wann welche Variante?"-Entscheidungshilfen

---

# Für Entwickler

---

Technische Spezifikationen, Design Tokens und CSS-Variablen.

## Quick Start

### Design Tokens einbinden

```
/* CSS-Variablen importieren */
@import 'design-system/build/css/variables.css';

/* Verwendung */
.button-primary {
  background-color: var(--button-primary-background-default);
  color: var(--button-primary-foreground-default);
}
```

### Token-Dateien

Format	Pfad	Verwendung
JSON (Quelle)	design-system/tokens/*.json	Token-Definitionen
CSS Variables	design-system/build/css/variables.css	Web-Projekte

#### Verfügbare Token-Dateien:

- colors.json - Farbpalette (Primär, Sekundär, Funktionale Farben)
- typography.json - Schriftsystem (Größen, Gewichte, Zeilenhöhen)
- spacing.json - Abstands-System (4px-Basis, 0-64px)
- buttons.json - Button-Tokens (Primary, Secondary, Tertiary)
- forms.json - Formular-Tokens (Input, Checkbox, Radio, Textarea, Select)
- navigation.json - Navigation-Tokens (Header, Breadcrumb, Footer, Drawer)
- feedback.json - Feedback-Tokens (Modal, Tooltip, Toast, Loading)

## Technische Referenz

### Technische Referenz-Dokumentation:

- [Design Tokens](#) - Token-Struktur, Verwendung, W3C DTCG Format
- [CSS-Variablen](#) - Vollständige Variablen-Referenz mit Kategorien
- [Grid & Breakpoints](#) - 12-Spalten-Grid, Responsive Breakpoints

### Token-Quelldateien:

- design-system/tokens/\*.json (Quelldateien)
- design-system/build/css/variables.css (Generierte CSS-Variablen)

## Grid & Breakpoints

Grid-System: 12 Spalten (definiert in `spacing-grid.md`)

Responsive Breakpoints:

```
/* Mobile: 320px - 767px */
/* Tablet: 768px - 1023px */
/* Desktop: 1024px+ */
```

Gutter-Spacing:

- Mobile: 16px (`var(--spacing-4)`)
- Tablet: 24px (`var(--spacing-6)`)
- Desktop: 32px (`var(--spacing-8)`)

Siehe: [Spacing & Grid Documentation](#)

## Komponenten-Spezifikationen

### Formulare

Jede Komponente dokumentiert States, ARIA-Attribute und Keyboard-Navigation:

- [Text-Input Specs](#)
- [Textarea Specs](#)
- [Select Specs](#)
- [Checkbox Specs](#)
- [Radio Button Specs](#)
- [Validation Patterns](#)

→ [Formular-Index](#)

Progressive Validation Pattern:

```
// onBlur initially, onChange after error
const [touched, setTouched] = useState(false);
const [error, setError] = useState(null);

const handleBlur = () => {
  setTouched(true);
  validate(value);
};

const handleChange = (e) => {
  setValue(e.target.value);
  if (touched) validate(e.target.value); // Real-time after first error
};
```

### Navigation

Focus-Management und ARIA-Patterns:

- [Header](#) - Sticky Desktop, Logo + Nav Links

- [Mobile Drawer](#) - Focus Trap, ESC-Handling, Transform Animation
- [Breadcrumb](#) - aria-current, Chevron-Separator
- [Footer](#) - CSS Grid, Multiple nav landmarks

→ [Navigation-Index](#)

**Focus Trap Pattern (Modal/Drawer):**

```
// Focus moves to first interactive element on open
// Tab wraps within container
// ESC closes (WCAG 2.1.2 No Keyboard Trap)
```

## Feedback

Modal-Patterns, Toast-Timing, Loading States:

- [Modal \(Radix UI recommended\)](#)
- [Toast \(Sonner recommended\)](#)
- [Tooltip \(Floating UI\)](#)
- [Loading States](#)

→ [Feedback-Index](#)

**Toast Auto-Dissmiss Timing:**

```
{
  success: 3000, // 3s - Quick confirmation
  info: 4000,    // 4s - More read time
  warning: 5000, // 5s - Important
  error: 0       // Never auto-dissmiss (requires acknowledgment)
}
```

## Buttons

- [Button Component](#) - Variants, Sizes, Icon Integration

**Button States:** default, hover, active, focus, disabled **Sizes:** small (32px), medium (40px), large (48px) **Variants:** primary, secondary, tertiary

## Accessibility Implementation

### WCAG 2.1 AA Compliance

Kritische ARIA-Patterns:

Komponente	Pattern	Beispiel
Modal	role="dialog", aria-modal="true", Focus Trap	<div role="dialog" aria-modal="true">
Toast	role="status" (success/info), role="alert" (warning/error)	<div role="alert">
Breadcrumb	aria-current="page" auf aktuellem Element	<a aria-current="page">
Mobile Drawer	aria-expanded, aria-controls, ESC zum Schließen	<button aria-expanded="false" aria-controls="drawer">
Checkbox/Radio	required, aria-required, aria-invalid, aria-describedby	<input aria-describedby="error-msg">

#### Touch Targets:

- Minimum: 44x44px (WCAG 2.5.5 Level AAA)
- Desktop compact: 32px acceptable for dense interfaces

#### Focus Indicators:

- 2px outline + 2px offset ( :focus-visible )
- Color: var(--hydrophon-blau-300) (3:1 contrast)

#### Color Contrast:

- Text: 4.5:1 minimum (WCAG 1.4.3)
- UI Components: 3:1 minimum (WCAG 1.4.11)

## Testing Guide

Vollständige Anleitung für Accessibility-Tests: → [Accessibility Testing Guide](#) - axe DevTools, Keyboard-Testing, Screen Reader Testing

#### Schnelltest-Checkliste:

1. Keyboard-Navigation: Alle interaktiven Elemente erreichbar via Tab
2. Focus-Indikatoren: Sichtbar bei :focus-visible
3. Screen Reader: VoiceOver/NVDA kann alle Elemente ankündigen
4. Color Contrast: Browser DevTools Contrast Checker
5. Touch Targets: Minimum 44x44px auf mobilen Geräten

## Build-Prozess

### Style Dictionary

#### Installation:

```
npm install style-dictionary
```

#### Token kompilieren:

```
cd design-system
npm run build
```

**Ausgabe:**

- design-system/build/css/variables.css - CSS Custom Properties

**Konfiguration:** design-system/config.json**Token-Struktur**

```
design-system/tokens/
└── colors.json      # Farbpalette (50-900 Abstufungen)
└── typography.json  # Schriften (Inter), Größen, Gewichte
└── spacing.json     # 4px-Basis-Abstände (0-64px)
└── buttons.json     # Button-Tokens (Primary/Secondary/Tertiary)
└── forms.json       # Input, Checkbox, Radio, Textarea, Select
└── navigation.json  # Header, Breadcrumb, Footer, Mobile Drawer
└── feedback.json    # Modal, Tooltip, Toast, Spinner, Progress
```

**Token-Format (W3C DTCG):**

```
{
  "color": {
    "primary": {
      "$value": "#0066cc",
      "$type": "color"
    }
  }
}
```

**CSS-Variablen Verwendung****Generated Output:**

```
:root {
  --color-primary: #0066cc;
  --button-primary-background-default: var(--color-primary);
}
```

**In Komponenten:**

```
.my-button {
  background-color: var(--button-primary-background-default);
  padding: var(--button-padding-medium);
  border-radius: var(--button-border-radius);
}
```

**Empfohlene Libraries****UI Primitives:**

- Radix UI (Modal, Tooltip): Battle-tested accessibility, headless styling

- **Floating UI (Tooltip positioning):** Smart viewport collision detection

**Notifications:**

- **Sonner (Toast):** Modern toast library, pause-on-hover, ARIA live regions

**Loading:**

- **react-loading-skeleton:** Auto-sizing skeleton screens, prevents layout shift

**Icons:**

- **Lucide React:** 1000+ icons, 2px stroke, customizable, tree-shakeable

## Komponenten-Dokumentation

Jede Komponente enthält:

1. **Übersicht** - Wann verwenden, visuelle Beispiele
2. **Varianten** - Alle verfügbaren Optionen
3. **Technische Spezifikation** - States, Tokens, Maße
4. **Accessibility** - ARIA-Patterns, Keyboard Navigation, Screen Reader
5. **Verwendungsbeispiele** - Code-Snippets
6. **Do's and Don'ts** - Best Practices

**Alle Komponenten-Dokumente:**

- [Foundation](#) | [Typography](#) | [Spacing](#) | [Effects](#) | [Icons](#)
- [Buttons](#)
- [Forms](#): Text Input, Textarea, Select, Checkbox, Radio, Labels, Validation
- [Navigation](#): Header, Mobile Drawer, Breadcrumb, Footer
- [Components](#): Product Card, Content Card, Data Table
- [Feedback](#): Modal, Tooltip, Toast, Loading

---

**Zielgruppe:** Frontend-Entwickler, Full-Stack Engineers **Zweck:** Schneller technischer Einstieg mit Code-Beispielen, Token-Referenz und ARIA-Patterns