HTML Tag 5: Barrierefreiheit im Web

Barrierefreiheit (Accessibility) im Web ist entscheidend, um sicherzustellen, dass Menschen mit Behinderungen gleichberechtigten Zugang zu Webseiten und Webanwendungen haben. Dies umfasst Menschen mit Sehbehinderungen, Hörbehinderungen, motorischen Einschränkungen oder kognitiven Behinderungen.

Die wichtigsten Webstandards für Accessibility stammen aus den Web Content Accessibility Guidelines (WCAG)

Grundlegende Prinzipien der Web-Accessibility

1. **Wahrnehmbarkeit**: Inhalte müssen für alle Sinne zugänglich gemacht werden.
2. **Bedienbarkeit**: Alle Benutzer müssen in der Lage sein, die Webseite zu bedienen.
3. **Verständlichkeit**: Inhalte müssen leicht zu verstehen sein.
4. **Robustheit**: Inhalte müssen mit einer Vielzahl von Hilfsmitteln funktionieren.

---

1. Semantische HTML-Struktur

Semantische HTML-Elemente wie `<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, `<footer>` und andere beschreiben den Inhalt besser und erleichtern Menschen, die auf Screenreader angewiesen sind, das Navigieren.

Beispiel:

```
<header>
  <h1>Meine Webseite</h1>
</header>
<nav>
  <ul>
    <li><a href="#home">Home</a></li>
    <li><a href="#about">Über uns</a></li>
  </ul>
</nav>
<main>
  <article>
    <h2>Artikelüberschrift</h2>
    <p>Inhalt des Artikels</p>
  </article>
</main>
<footer>
  <p>&copy; 2024 Meine Webseite</p>
</footer>
```

Vorteil: Semantische HTML-Tags geben Screenreadern und Suchmaschinen klare Hinweise über die Bedeutung der Inhalte.

---

2. Alternative Texte für Bilder

Bilder müssen mit dem Attribut `alt` versehen sein, um eine Textalternative für Screenreader und Benutzer mit Sehbehinderungen bereitzustellen.

Beispiel:

```
<img src="katze.jpg" alt="Ein süßes Kätzchen, das auf dem Fensterbrett schläft.">
```

Tipp: Der `alt`-Text sollte den Inhalt und Zweck des Bildes beschreiben.

3. Formulare zugänglich gestalten

Formulare müssen richtig gekennzeichnet und strukturiert sein, um sie für alle Benutzer zugänglich zu machen.

Verwendung von `<label>`:

Jedes `<input>`-Element sollte mit einem `<label>` verknüpft sein, um den Zweck des Feldes anzugeben.

```
<form>
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required>

  <label for="email">E-Mail:</label>
  <input type="email" id="email" name="email" required>

  <button type="submit">Absenden</button>
</form>
```

Barrierefreie Formularstrukturen:

- **Verwende `aria-*`-Attribute** (z.B. `aria-label`, `aria-describedby`), um zusätzliche Informationen für Screenreader bereitzustellen.
- **Füge `required`-Attribute** hinzu, um Benutzer klar zu machen, welche Felder erforderlich sind.

---

4. Farbkontrast und visuelle Zugänglichkeit

Stellen Sie sicher, dass der **Kontrast zwischen Text und Hintergrund** groß genug ist, damit Text für Benutzer mit Sehschwächen lesbar ist.

WCAG-Empfehlungen:

- Der Kontrast sollte mindestens **4.5:1** für normalen Text und **3:1** für großen Text betragen.

Beispiel für CSS:

```
body {
  background-color: #ffffff;
  color: #000000;
}
```

Vermeiden Sie reine Farbunterscheidungen (z.B. in Diagrammen oder Links) ohne zusätzliche visuelle Hinweise wie **Unterstreichungen**.

---

5. Navigationshilfen

- **Verwende Skip-Links**: Ermögliche es Nutzern, direkt zum Hauptinhalt zu springen.

```
<a href="#maincontent" class="skip-link">Zum Inhalt springen</a>
```

- **Tastaturnavigation**: Alle Funktionen sollten mit der Tastatur erreichbar sein. Achte darauf, dass interaktive Elemente wie Links, Schaltflächen und Formulare korrekt fokussierbar sind.

6. Überschriftenhierarchie

Überschriften sollten in logischer Reihenfolge (`<h1>`, `<h2>`, `<h3>`) verwendet werden, um eine klare Struktur zu schaffen. Diese Struktur hilft Screenreadern, den Inhalt besser zu verstehen und zu navigieren.

Beispiel:

```
<h1>Hauptüberschrift</h1>
<h2>Unterüberschrift</h2>
<h3>Abschnittsüberschrift</h3>
```

---

7. Video- und Audiocontent

Multimedia-Inhalte sollten barrierefrei sein, indem **Untertitel**, **Transkripte** oder **Beschreibungen** hinzugefügt werden.

Video mit Untertiteln:

```
<video controls>
  <source src="video.mp4" type="video/mp4">
  <track src="untertitel.vtt" kind="subtitles" srclang="de" label="Deutsch">
</video>
```

Tipp: Textbeschreibungen sollten für blinde Benutzer bereitgestellt werden, z.B. über das `track`-Element.

---

8. Fokusmanagement

Interaktive Elemente (z.B. Links, Schaltflächen) sollten in einer sinnvollen Reihenfolge durch Fokussieren (Tab-Reihenfolge) erreichbar sein.

Fokus-Highlighting:

```
button:focus {
  outline: 2px solid blue;
}
```

---

9. Verantwortungsvolle Nutzung von Animationen und Effekten

Bewegungen und Animationen können einige Benutzer negativ beeinflussen. Verwende Animationen sparsam und biete Optionen zum Deaktivieren oder Reduzieren an (z.B. durch die Nutzung von `prefers-reduced-motion` in CSS).

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

10. ARIA

ARIA (Accessible Rich Internet Applications) ist eine Spezifikation, die Entwicklern hilft, interaktive und dynamische Webanwendungen zugänglicher zu machen, insbesondere für Menschen, die Screenreader oder andere assistive Technologien nutzen. ARIA fügt zusätzliche Informationen zur Barrierefreiheit hinzu, indem es HTML um spezielle Attribute erweitert, die den Zustand und die Struktur von Inhalten und Benutzeroberflächen genauer beschreiben.

ARIA-Rollen

Eine Rolle gibt an, wie ein Element auf einer Webseite behandelt werden soll. Diese Rollen helfen assistiven Technologien wie Screenreadern, den Benutzern klar zu machen, welche Funktion ein bestimmtes Element auf der Seite hat, auch wenn das Standard-HTML-Element selbst dies nicht ausreichend ausdrückt.

Banner: Wird für ein übergeordnetes Element verwendet, das den Banner-Inhalt einer Seite darstellt, normalerweise im Header.

```
<header role="banner">Website Header</header>
```

Navigation: Gibt an, dass ein Element eine Navigationsleiste enthält.

```
<nav role="navigation">...</nav>
```

Main: Kennzeichnet den Hauptinhalt der Seite, sodass Benutzer diesen schneller finden können.

```
<main role="main">...</main>
```

Button: Eine explizite Rolle für Elemente, die wie Schaltflächen agieren, jedoch keine `<button>`-Elemente sind (z. B. `<div>` oder `<span>`).

```
<div role="button" tabindex="0">Klick mich!</div>
```

Alert: Wird verwendet, um dynamische Inhalte, wie Warnungen oder Fehlermeldungen, hervorzuheben.

```
<div role="alert">Fehler: Ungültige Eingabe!</div>
```

… weitere ARIA Roles

ARIA-Attribute

ARIA-Attribute bieten zusätzliche Informationen über den Zustand und das Verhalten von interaktiven oder dynamischen Elementen auf der Seite. Sie helfen, Inhalte und deren Zustände für assistive Technologien zu beschreiben.

aria-label: Fügt einem Element eine Textbeschreibung hinzu, die von assistiven Technologien wie Screenreadern gelesen wird, wenn keine sichtbare Beschriftung vorhanden ist.

```
<button aria-label="Suche starten">🔍</button>
```

aria-expanded: Gibt an, ob ein UI-Element wie ein Akkordeon oder ein Dropdown-Menü erweitert oder eingeklappt ist.

```
<div id="dialog" role="dialog" aria-labelledby="dialog-title">
  <h2 id="dialog-title">Bestätigungsdialog</h2>
```

```
</div>
```

aria-hidden: Signalisiert assistiven Technologien, dass ein Element derzeit nicht relevant ist und ignoriert werden soll.

```
<div aria-hidden="true">Nicht sichtbarer Text</div>
```

aria-live: Bestimmt, wie dynamische Inhalte behandelt werden sollen. Zum Beispiel kann eine Region, die mit aria-live="polite" gekennzeichnet ist, eine nicht dringende Aktualisierung anzeigen, ohne den Benutzer zu unterbrechen.

```
<div aria-live="polite">Aktualisierte Nachricht: Ihr Formular wurde gesendet.</div>
```

... weitere ARIA Attribute