Jak tworzyć dostępny kod?

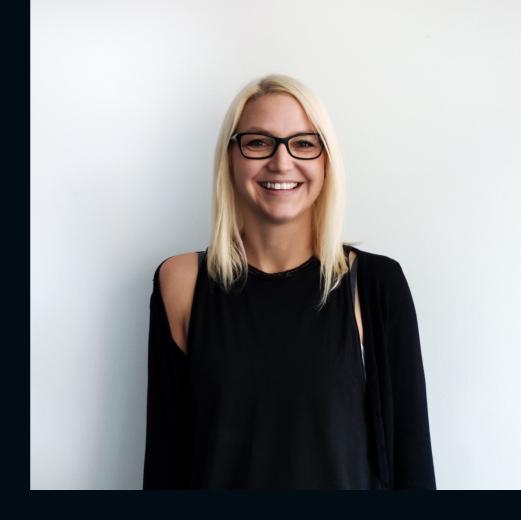
Praktycznie dla Frontendowców

Ania Karoń

Specjalista ds. Dostępności i Senior Frontend Developer Snowdog

☑ anna.karon@snow.dog

in Linkedin



Spis treści

- 1 / Frontendowa piaskownica
- 2 / Struktura i semantyka
- 3 / Nagłówki
- 4 / Dostępna nazwa
- 5 / Jak skutecznie schować element?
- 6 / Focus
- 7 / ARIA najczęściej
- <mark>8 /</mark> W praktyce
- 9 / Axe devTools
- 10 / Checklista A11y dla frontendowca
- 11 / Polecajki Testowanie dostępności
- 12 / Polecajki różne

Frontendowa piaskownica

- HTML
- CSS
- JavaScript
- WAI-ARIA (Web Accessibility Initiative Accessible Rich Internet Applications)

Struktura i semantyka

- Dokument HTML RWD, język, tytuł
- Landmarks: `header`, `main`, `footer`,
 `section`, `nav`, `aside`
- Semantyczne elementy HTML: ``, `
 `, `<button>`, ``, `<article>`, `<address>`,
 `<time>`...

```
<button type="button">
   Good button
</button>

<div @onclick="close" role="button">
   Bad button
</div>
```

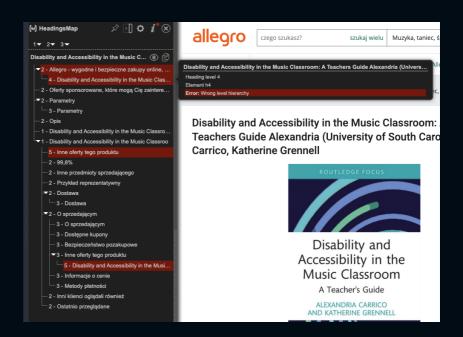
```
<html lang="en">
```

Nagłówki

Nagłówki - `<h1>` - `<h6>`

- kolejność nagłówków
- semantyka nad stylami

HeadingsMap - rozszerzenie do przeglądarki (Chrome, Firefox)



Dostępna nazwa

Niektóre elementy otrzymują swoje nazwy od treści, na przykład: przyciski, linki i nagłówki. Najlepszym sposobem jest użycie widocznej nazwy, która jest spójna dla wszystkich użytkowników.

Pełną listę elementów, które otrzymują nazwę od swojej zawartości, można znaleźć w dokumantach ARIA: Nadawanie nazw na podstawie zawartości podrzędnej

Natywne znaczniki do nazywania elementów HTML:

- `label` dla pól formularzy: `input`, `textarea`, `select`
- `legend` dla elementu `fieldset`
- `caption` dla elementu `table`
- `figcaption` dla `figure`
- **-** ...

Elementy graficzne bez widocznej etykiety powinny mieć tekst alternatywny

Nazwy alternatywne

aria-label

Dodajemy bezpośrednio do elementu

```
<!-- overrides the content -->
<button type="button" aria-label="Close">
    X
</button>

<!-- name and content are accessible -->
<nav aria-label="Customer">

            <a href="/profile">Profile</a>
            <a href="/settings">Settings</a>
            <a href="/logout">Logout</a>

</nav>
```

aria-labelledby

Podpinamy element (np. nagłówek), który opisuje sekcje używając jego `id`

```
<h2 id="penguins-heading">Penguins list</h2>

        King penguin
        Adelie penguin
        Gentoo penguin
```

aria-labelledby docs

aria-label docs

Warto pamiętać

- Nie nadużywaj ARIA! Nie każdy element potrzebuje `aria-label`. Jeśli to możliwe, używaj natywnego rozwiązania HTML.
- Nie używaj 'aria-label' na elementach 'div', 'span' i 'paragraph'. Sprawdź czy dany atrybut lub rola
 ARIA są dozwolone na danym elementcie.
- Gdy używasz `aria-label` lub `aria-labelledby` na przycisku lub linku z widocznym tekstem, wartość nazwy powinna odpowiadać widocznemu tekstowi lub przynajmniej zaczynać się od widocznego tekstu (WCAG SC 2.5.3). Jest to ważne dla użytkowników korzystających z poleceń głosowych, którzy odwołują się do widocznych etykiet.

Więcej informacji: Providing Accessible Names and Descriptions

Alternatywna treść, uzupełnienie

aria-describedby

Używamy, by dodać dodatkowy opis (np. do pola formularza lub przycisku), podpinamy element, używając jego id

```
<form>
  <label for="fname">First name</label>
  <input aria-describedby="int2" autocomplete="given-name" id="fname" type="text">
  Your first name is sometimes called your "given name".
</form>
```

aria-describedby docs

Jak skutecznie schować element?

Warto pamiętać! Ukrywaj z rozwagą, może to wpłynąć na SEO (np, wizualne ukrycie nagłówka)

Dla wszystkich

```
.element {
  display: none;
  visibility: hidden;
}
```

Dla technologii asystujących, nie wizualnie

```
`aria-hidden="true"`
<span aria-hidden="true">Decorative element</span>
```

Tylko wizualnie

```
/* "sr" meaning "screen-reader" */
.sr-only:not(:focus):not(:active) {
  clip: rect(0 0 0 0);
  clip-path: inset(50%);
  height: 1px;
  overflow: hidden;
  position: absolute;
  white-space: nowrap;
  width: 1px;
}
```

TailwindCss - `sr-only`

Focus

stan `focus` powinien być widoczny

- css `outline`, przeglądarki dostarczają `focus`, jeśli nie dodajecie swoich styli, nie ustawiajcie `outline:
 none`
- kolejność elementów fokusowalnych
- :focus`i`:focus-visible`

ARIA najczęściej

Live region

- `aria-live` wskazuje AT dynamicznie zmieniające się na stronie (ustawiamy na kontenerze)
- wartość `polite` informuje o zmianie, kiedy akończone zostanie aktualne zadanie
- wartość `assertive` informuje o zmianie natychmiast
- `aria-atomic` informuje, jak elementy w "obserwowanym" regionie (oznaczonym atrybutem `aria-live`) mają zostać przedstawione tylko zmiany: `false`, czy wszystkie elementy: `true`

Zmiana widoczności elementu

aria-expanded="true / false"`- informuje o tym czy element jest widoczny czy schowany

Kitties list

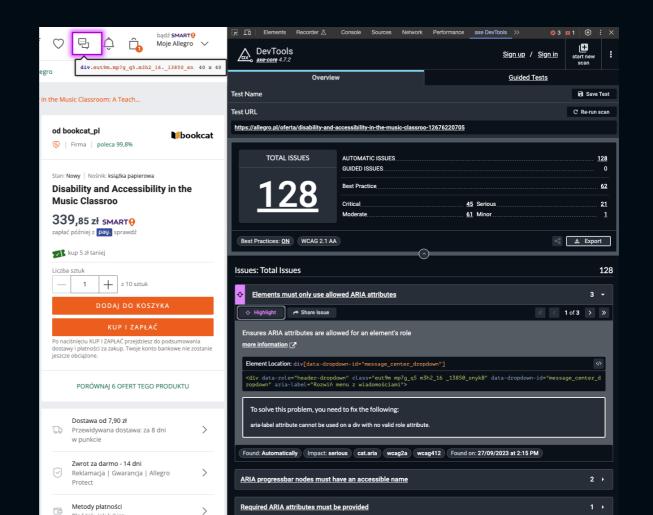
Kitten Rocky



☐ With sister **(i)**

Axe devTools

Axe DevTools - rozszerzenie do przeglądarki



Checklista A11y dla frontendowca

- 1 / Uruchom Axe DevTools w przeglądarce, sprawdź błędy i napraw je. Dobrze jest mieć 0 błędów (chyba że problem nie jest związany z funkcjonalnością, nad którą pracujesz lub został naprawiony w innym zadaniu).
- 2 / Sprawdź kolejność nagłówków na całej stronie (czy nowy komponent nie narusza tej kolejności). Możesz użyć rozszerzenia przeglądarki HeadingsMap
- 3 / Jeśli pracujesz nad komponentem, który pokazuje i ukrywa zawartość (na przykład listę rozwijaną, dialog, dropdown itp.), uruchom skanowanie Axe DevTools dla obu wersji komponentu widocznej i ukrytej.
- 4 / Sprawdź wszystkie przyciski, które otwierają ukrytą zawartość powinny mieć atrybut `aria-expanded="true/false"`.
- 5 / Sprawdź wszystkie przyciski, które otwierają dialogi /popupy powinny mieć atrybuty `aria-expanded="true/false"` i `aria-haspopup="dialog"`.
- 6 / Przetestuj stronę klawiaturą i sprawdź czy:
 - wszystkie funckjonalności obsługiwane za pomocą myszy są również obsługiwane za klawiaturą
 - wszystkie interaktywne elementy są fokusowalne
 - stan `focus` jest widoczny i wyróżniający

Polecajki - Testowanie dostępności

Testy automatyczne

- Axe DevTools- rozszerzenie do przeglądarki
- Wave rozszerzenie do przeglądarki
- Lighthouse
- IBM Accessibility Equal Access Accessibility Checker

Testy manualne

- sprawdzenie kodu
- ANDI
- Czytnik ekranu VoiceOver (macOS), Narrator lub NDVA (Windows), ChromeVox (rozszerzenie do przeglądarki)

Polecajki różne

- Accessibility Non-checklist
- HeadingsMap chrome extension
- Blog post mojego autorstwa o Accessible Name
- Providing Accessible Names and Descriptions
- ARIA Authoring Practice Guide Patterns
- CodePen Kitties

Dziękuję bardzo!

Ania Karoń

☑ anna.karon@snow.dog

in Linkedin