WCAG NA PRZYKŁADZIE FRAGMENTU STRONY YOON GROUP

SPIS TREŚCI

PODSTAWOWE POJĘCIA

CZYTNIK EKRANOWY

Program, który przetwarza tekst na mowę. Dzięki niemu osoba niewidoma lub niedowidząca może zapoznać się z treścią na stronie oraz opisem grafik. W przypadku filmów pozwala przeczytać napisy do niego.

Należy tu zaznaczyć, że czytniki ekranowe są bardzo różne i różnie działają, od darmowych jak wtyczka do chrome Screen Reader i program do pobrania NVDA po JAWS, który kosztuje ponad 4 tys. złotych.

Przykładowa różnica między nimi jest taka, że ten pierwszy w przypadku poprawnie zakodowanej tablicy przeczyta ją jednym ciągiem, a ten ostatni będzie dokładnie informował użytkownika w której kolumnie i wierszu się znajduje.

W Polsce najbardziej popularna jest NVDA, która działa na Windows oraz czytnik wbudowany w McBook. NVDA najlepiej działa na FireFox, dlatego w przypadku kodowania stron jest ważne, aby sprawdzać, czy tam nasz kod działa dobrze.

Z tego co wiem, to pod Linux obecnie nie ma żadnego dobrego czytnika ekranowego.

PODSTAWOWY KOD

KOD WIDOCZNY DLA CZYTNIKÓW EKRANOWYCH

Poniższy kod jest niezbędny jeżeli mówimy o kodowaniu dostępnej strony, ponieważ pozwala na ukrywanie elementów dla użytkownika „widomego” i przekazywanie dodatkowych informacji dla użytkownika „niewidomego”.

```
.sr-only {
  border: 0 !important;
  clip: rect(1px, 1px, 1px, 1px) !important;
  -webkit-clip-path: inset(50%) !important;
  clip-path: inset(50%) !important;
  height: 1px !important;
  margin: -1px !important;
  overflow: hidden !important;
  padding: 0 !important;
  position: absolute !important;
  width: 1px !important;
  white-space: nowrap !important;
}

.sr-only-focusable:focus,
.sr-only-focusable:active {
  clip: auto !important;
```

```
  -webkit-clip-path: none !important;
  clip-path: none !important;
  height: auto !important;
  margin: auto !important;
  overflow: visible !important;
  width: auto !important;
  white-space: normal !important;
}
```

Z tego co pamiętam, to Bootstrap też ma swoje klasy pod to, ale nie pamiętam ich nazw.

KOD UKRYWAJĄCY ELEMENTY DLA CZYTNIKÓW EKRANOWYCH

Istnieje coś takiego jak aria-hidden=”true”, która działa tak, że ukrywa dane elementy dla czytników ekranowych.

Przykład z Yoon Group:

```
<button class="language__btn language__btn--black">
    <span class="sr-only" lang="pl">zmień język na polski</span>
    <span aria-hidden="true">Pl</span>
</button>
```

Opiszę ten kod: w nim za pomocą sr-only przekazałam informację dla czytnika, aby przeczytał całość „zmień język na polski” oraz zablokowałam dla niego wymówienie „PL”. W ten sposób przycisk jest bardziej zrozumiały dla osoby korzystającej z czytnika, która by musiała się domyślić, co oznacza samo „PL”.

Aria-hidden jest przydatne również przy ikonkach, których nie chcemy przekazywać do czytnika, bo często nie ma potrzeby, aby przeczytał „uśmiechnięta buźka” (czytnik z chrome tak czyta te emotki: 😊).

BARDZO WAŻNE! Jeżeli nie znasz zasad dotyczących aria-hidden, nie czujesz się w nich bardzo pewnie, to go NIGDY nie używaj. Istnieje duża szansa, że zablokujesz dla czytnika jakąś bardzo ważną informację.

KOD PRZY KODOWANIU PLIKU Z PSD

Dostępna strona, to responsywna strona i powinna się wyświetlać dobrze przy powiększeniu do 200%.

Poniżej kod, który pomaga przy kodowaniu stron z pliku PSD, Figma, itp.:

```
body {
  font-size: 62.25%;
}
```

Gdzie:

1rem = 10px

1.6rem = 16px

JĘZYK STRONY I ELEMENTÓW

Czytnik ekranowy do poprawnego działania potrzebuje takich informacji, jak język strony lub elementu. Oznacza się go za pomocą atrybutu „lang”.

DLACZEGO JEGO UŻYWANIE JEST WAŻNE?

Jeżeli język strony zostanie ustawiony na angielski (html lang=”en”), to będzie to informacja dla czytnika, że musi przełączyć się na ten język. Bez tej informacji będzie próbował przeczytać całość w języku na jaki został ustawiony przez użytkownika, np. po polsku, co może dać dziwny efekt. To samo dotyczy elementów na stronie, gdzie również trzeba powiedzieć czytnikowi „hej, bo my tu mamy słowo po polsku, przeczytaj go po polsku, nie po angielsku, okej?”, np. Wrocław

PROBLEMY PRZY KOMERCYJNYCH STRONACH

Jeżeli robicie strony pod CMS, gdzie użytkownik sam dodaje treść to zwykle nie będziecie mieli wpływu na ustawienie języka pod treść. Przeciętny użytkownik w kodzie też tego raczej sam sobie nie ustawi.

TEKST ALTERNATYWNY

Tekst alternatywny (atrybut alt) dla grafik jest bardzo ważny, ponieważ pozwala sprawić, że użytkownik „niewidomy” zobaczy obraz.

PODSTAWOWE ZASADY

- Powinien możliwe najlepiej opisywać obraz, np. uśmiechnięty mężczyzna na czerwonym rowerze jedzie po ruchliwej ulicy
- Powinien zawierać kolory, ponieważ osoba niedowidząca je często zna. Mitem jest, że jeżeli ktoś nie widzi, to nie zna kolorów
- W przypadku logotypów spotkałam się z opinią, że dobrze jest dodać opis jak dane logo wygląda, ale tu każdy mówi inaczej
- Jeżeli na jednej grafice jest kilka różnych logotypów, to powinno się rozdzielić grafikę (jeżeli się da), tak aby każde logo było osobno i miało swój własny tekst alternatywny
- Spotkałam się z opinią, że tekst alternatywny powinien zawierać max 100 znaków, ponieważ niektóre czytniki go ucinają po tej długości. Jeżeli tekst jest za długi, to dobrze dodać opis do zdjęcia, choćby ukryty za pomocą sr-only
- Jeżeli grafika jest tylko ozdobą, np. jest to linia, to należy dodać pusty alt (alt=””), w innym razie czytnik przeczyta nazwę pliku, a z pustym alt po prostu pominie grafikę
- Infografiki zawsze muszą zostać opisane, to samo plakaty

PROBLEMY Z TEKSTEM ALTERNATYWNYM

W przypadku komercyjnych stron za grafiki najczęściej odpowiada klient, więc nie mamy możliwości kontrolowania, czy alt został dodany poprawnie. Zwłaszcza, że nawet zajmując się tematem dostępności to ułożenie poprawnego alt jest trudne. Dlatego dobrze jest zaprogramować stronę tak, aby w najgorszym przypadku przy każdym zdjęciu zawsze był dodawany pusty alt. Czasem lepiej aby czytnik nic nie przeczytał, niż przeczytał xsnakppaodano.png

ROZMIAR PLIKÓW

Jeżeli na stronie pojawia się możliwość pobrania czegoś, np. obrazu, pliku, to obok linku pobierania powinna pokazać się również wielkość pliku.

SKIP LINKI

Skip linki pozwalają na „przeskakiwanie” do interesującej nas treści. Podstawą jest dodanie przynajmniej jednego przed nawigacją, na samej górze strony. Maksymalnie może być ich około 4-5, więcej się nie zaleca. Jednak skip linki mogą pojawić się również tam, gdzie na stronie dodaje się tablice, tak aby łatwiej przez nie przejść (bez konieczności przeklikania ich i tracenia czasu).

DLACZEGO SĄ WAŻNE?

Użytkownik poruszający się za pomocą tabulatora bez nich musi przejść kolejno po każdym elemencie, co często zajmuje dużo czasu. Użytkownik korzystający z czytnika ekranowego dzięki nim przy każdym wejściu na stronę nie musi za każdym razem odsłuchiwać całej nawigacji, jeżeli to ma być strona, gdzie często ktoś wchodzi, np. z informacjami, to skip linki mogą umilić komuś korzystanie ze strony.

KOD POD SKIP LINKI

HTML:

```html
<ul class="skip-links wrapper">
  <li><a href="#main" class="skip-link">Go to main content</a></li>
  <li><a href="#contact" class="skip-link">Go to contact</a></li>
</ul>
```

SCSS:

```scss
.skip-links {
  list-style: none;
}

.skip-link {
  clip: rect(1px, 1px, 1px, 1px) !important;
  -webkit-clip-path: inset(50%) !important;
  clip-path: inset(50%) !important;
  height: 1px !important;
  margin: -1px !important;
  overflow: hidden !important;
  position: absolute !important;
  width: 1px !important;
  white-space: nowrap !important;
  text-align: center;
  text-decoration: none;
  position: absolute;
  top: 1rem;
  left: 1rem;
  z-index: 999;
```

```
  color: black;
  background: white;
  border: 2px dotted black;
  padding: 0.5rem 1.5rem;
  &:focus,
  &:active {
    clip: auto !important;
    -webkit-clip-path: none !important;
    clip-path: none !important;
    height: auto !important;
    margin: auto !important;
    overflow: visible !important;
    width: auto !important;
    white-space: normal !important;
  }
}
```

Ten kawałek kodu:

```
  color: black;
  background: white;
  border: 2px dotted black;
  padding: 0.5rem 1.5rem;
```

może być zmieniany zależnie od projektu, tak aby pasował do całości strony. Nie trzeba też tam koniecznie ustawiać obramowania 2px kropki, ale w profesjonalny audycie, który czytałam ktoś właśnie to zalecał, nie pisało tam dlaczego.

KOLORY NA STRONIE

Przeciętny daltonista nie rozróżnia koloru czerwonego, ale za to przeciętny daltonista rozróżnia kolor niebieski. Bardzo polecam to narzędzie do badania kontrastu: https://whocanuse.com/

W skrócie kontrast poniżej 4.5:1 (tekst do tła) jest często zbyt niski nawet dla osoby, która widzi dobrze. 7:1 jest zalecany, a 21:1 to maksymalny (czarny do białego). Są też osobne zasady do kontrastu dla obramowania przycisku, chyba 3:1.

NAGŁÓWKI

Zasady są takie:

1. Nazwa strony to zawsze h1.

2. Każda sekcja musi mieć swój nagłówek, gdzie najczęściej będzie to h2.

3. Jeżeli sekcja zawiera podsekcje, to one również muszą mieć swój nagłówek, czyli najczęściej h3.

4. Jeżeli na stronie pojawia się okno modalne, to jego kod powinien zostać zamieszczony pod stopką (footer) i nazwa tego okna od nowa zaczyna się od h1.

Jeżeli w szablonie od grafika nie ma widocznych nagłówków sekcji to one nadal muszą zostać dodane w kodzie, ale nie muszą być widoczne dla użytkownika "widzącego". W takim przypadku można skorzystać ze specjalnej klasy "sr-only".

PRZYKŁAD Z YOON GROUP

Tutaj dodałam taki kod:

```
<h1 class="sr-only wrapper">Yoon Group</h1>
```

ponieważ niżej pojawia się grafika z logiem:

```
<a href="#" class="logo">
    <img src="img/logo-yg.svg" alt="Yoon Group - main page" />
</a>
```

Ten „wrapper” przy h1 jest zbędny, ale dodałam go, aby w razie gdyby z jakiegoś powodu klasa sr-only nie zadziałała, to aby „Yoon Group” nie wyglądało bardzo źle.

DOSTĘPNE LINKI

Linki powinny być oznaczone kolorem oraz podkreślone, ale w Waszych stronach podkreślenie raczej rzadko będzie wchodzić w grę, bo jednak ono zmienia wygląd.

OTWIERANIE STRONY W NOWEJ KARCIE

Jeżeli strona otwiera się w nowej karcie to należy poinformować o tym użytkownika. Tak samo jeżeli otwiera się jako pdf (nie każdy czytnik działa z plikiem pdf).

Ogólnie ja spotkałam się z dwoma szkołami:

- Dodanie tooltipa z informacją (podobno jest spoko też dla osób starszych)
- Wykorzystanie klasy sr-only, np.

```
<a href="#" target="_blank">Privacy Policy<span class="sr-
only">open in a new tab</span></a>
```

PORUSZANIE SIĘ ZA POMOCA KLAWIATURY

Cała strona/wszystkie elementy klikalne powinny być możliwe do przejścia za pomocą klawiatury. Ważne jest przy tym pamiętanie, aby focus był dobrze widoczny. W przypadku używania reset.css należy samodzielnie bardziej zadbać o focus, bo z tego pamiętam to on go w ogóle całkowicie usuwa.

NAWIGACJA

W przypadku nawigacji należy pamiętać o tym, że jeżeli mamy nawigację rozwijaną to każdy element powinien być możliwy do przejścia za pomocą tabulatora. Spotkałam się też z opinią, że w przypadku takiej bardzo dobrze jest dodać możliwość poruszania się pomiędzy jej głównymi elementami za pomocą strzałek.

Sama jako bazy używam kodu od Tomasza Jakuta (Comandeer):
https://codepen.io/Comandeer/pen/aboLmaK

Oczywiście trzeba tu pamiętać, aby zawsze zmieniać „Otwórz/zamknij” zależnie od głównego języka strony.

JUSTOWANIE TEKSTU

Justowanie tekstu (od lewej do prawej) jest złą praktyką, ponieważ osoby z problemami poznawczymi mogą mieć problemy z przeczytaniem go. Zaleca się wyrównanie tekstu do lewej strony.

FORMULARZE

ETYKIETY

Każdy input musi mieć przypisany mu label. Z tego co pamiętam to placeholder nie jest odczytywany przez czytniki ekranowe. Nie zaleca się też programowania etykiet, które robią coś takiego, że najpierw wyglądają jak placeholder, a po kliknięciu w niego wjeżdżają nad pole.

WALIDACJA

Zalecana walidacja wygląda tak, że w przypadku błędu to informacja o nim powinna pojawić się pod polem wraz z ikonką wykrzyknika. Tekst chyba powinien być czerwony, a pole powinno mieć wtedy czerwone obramowanie. Dobrze jest jeżeli użytkownik zostanie od razu przeniesiony na dane pole z błędem.

INFORMACJA DLA OSÓB Z STANAMI LĘKOWYMI

Taki użytkownik musi otrzymać dokładną informację co się wydarzyło, np. formularz został wysłany lub co się wydarzy, np. przyjdzie do Pana/Pani/Ciebie e-mail na wskazany adres z potwierdzeniem informacji.

PROBLEMY Z CAPTCHA

Tutaj ważne jest aby dodawać taki, który ma możliwość odsłuchania tekstu.

WTYCZKI I KONTROLKI

Gotowe wtyczki to zło, jeżeli potrafisz programować to NIGDY ich nie używaj, ponieważ często nie są dostępne i są zaprogramowane po prostu źle. Kiedyś widziałam sytuację, gdzie dzięki takiej wtyczce przy zmianie kontrastu zdjęcie pielęgniarki wyglądało jakby była postacią z horroru.

Mówi się też, że dobrze zaprogramowana strona nie potrzebuje żadnych wtyczek.

Natomiast jeżeli chodzi o kontrolki to wersje są różne, bo to bardzo zależy od tego kim jest przeciętny użytkownik strony. Jedni odradzają, bo źle działają z oprogramowaniem którym posiłkują się osoby z niepełnosprawnością, a inni zalecają, bo np. powiększenie tekstu jest pomocne osobie starszej, która nie za dobrze zna się na komputerach.

PROBLEMY Z KONTROLKAMI

W przypadku tworzenia strony dla budżetówki może zdarzyć się sytuacja, gdzie chociaż strona została zrobiona prawidłowo i nie potrzebuje takich dodatków, ale urzędnik i tak będzie ich wymagał. Jest to

obecnie duży problem przy kodowaniu stron dla szkół. Niby można się kłócić z nim, ale chyba prościej w takim przypadku po prostu zaprogramować samodzielnie kontrolki.