



Tomasz Jakut

Przycisk i accessibility, czyli
o pozornej prostocie dostępności.

dev.js summit 2021

SALA CZERWONA

Godz. 10:00



BlockchainWares



divante

softserve



BRAINHUB



accenture

creativestyle



STXNEXT
python powerhour

CKSOURCE

intive

tellyo

DC DEVELOPER
COUNCIL



luxoft
A DAC Technology Company

Beesafe

HL TECH



EARTH CLASS MAIL

CRODU

PROMATIC
GROUP

searchspring

filestack



Infermedica

MACOPEDIA

Przycisk i accessibility

czyli o pozornej prostocie dostępności



O mojej skromnej osobie słów kilka

CKSource

Github: Comandeer

Twitter: Comandeer2

WebKrytyk.pl

Blog.Comandeer.pl

O czym będziemy mówić?

- Tworzenie dostępnego przycisku
- Nieoczywiste problemy przyciskowe
- Kilka innych wyzwań dostępnościowych
- Garść wskazówek

Przycisk

Large button

Large button

Large button

Large button

Wygląd

- Określony we WCAG 2.1 w 1.4.3/1.4.6 (Kontrast), w 1.4.11 (Kontrast elementów nietekstowych) oraz w 2.5.5 (Rozmiar celu dotykowego).
- Kontrast między tekstem na przycisku a jego tłem powinien wynosić 4.5:1/7:1.
- Kontrast między przyciskiem a otaczającą go treścią powinien wynosić 3:1.
- Przycisk powinien mieć wymiary co najmniej 44x44 piksele.

WCAG

- Standard określający podstawowe zalecenia odnośnie dostępności.
- Podzielony na trzy poziomy:
 - A
 - AA
 - AAA
- Podzielony na zalecenia i kryteria sukcesu
- Zawiera listę proponowanych rozwiązań: <https://www.w3.org/WAI/WCAG21/Techniques/>
- Zawiera dokładne tłumaczenia zaleceń:
<https://www.w3.org/WAI/WCAG21/Understanding/>
- Polska wersja: <https://wcag21.lepszyweb.pl/>
- Oryginalna wersja: <https://www.w3.org/TR/WCAG21/>

Jak sprawdzić kontrast?

- Obliczyć ze wzoru:
<https://www.w3.org/WAI/WCAG21/Techniques/general/G18#tests>
- <https://whocanuse.com/>
- <https://contrast-ratio.com/>
- Lighthouse, aXe i inne narzędzia automatyczne

Identyfikacja

- Określona we WCAG 2.1 w 1.3.1 (Informacje i relacje) oraz w 1.3.6 (Określenie przeznaczenia).
- Niestandardowy przycisk musi być poprawnie przedstawiony w drzewku dostępności.
- Można użyć atrybutu `[role=button]` z ARIA.

Drzewko dostępności – Chrome

The screenshot displays the Chrome DevTools interface with the Accessibility panel open. The left pane shows the DOM tree with the following structure:

```
<!DOCTYPE html>
<html lang="pl" prefix="og: http://ogp.me/ns#" vocab="http://schema.org">
  <head>...</head>
  <body screen_capture_injected="true">
    <header class="site-header">...</header>
    <main class="page-content">
      <div class="wrapper">
        <article class="post" typeof="BlogPosting">
          <header class="post-header">
            <h2 class="post-title" property="name headline">Tworzmy czytnik
            ekranowy</h2>
            <p class="post-meta">...</p>
          </header>
          <div class="post-content" property="articleBody">...</div>
          <div id="disqus_thread">...</div>
          <script>...</script>
          <noscript>...</noscript>
        </article>
      </div>
    </main>
  </body>
</html>
```

The right pane shows the Accessibility tree for the selected `h2` element:

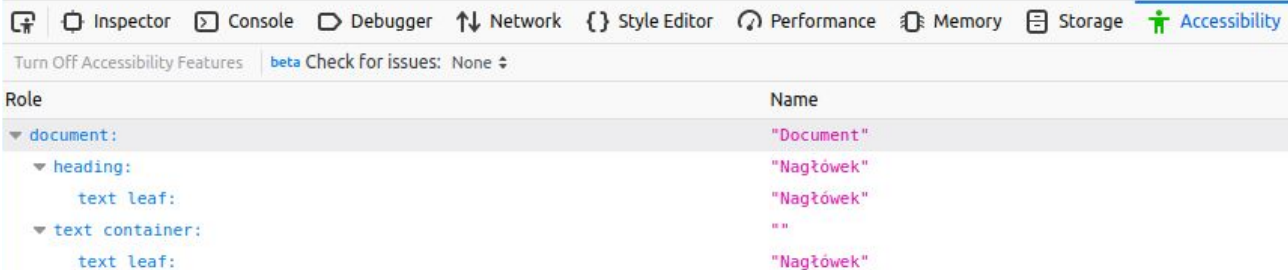
```
main
  article
    group
      heading "Tworzmy czytnik ekranowy"
        text "Tworzmy czytnik ekranowy"
    ARIA Attributes
      No ARIA attributes
    Computed Properties
      Name: "Tworzmy czytnik ekranowy"
        aria-labelledby: Not specified
        aria-label: Not specified
        Contents: "Tworzmy czytnik ekranowy"
        title: Not specified
      Role: heading
      Level: 2
```

The breadcrumb at the bottom of the DOM tree is: `html > body > main.page-content > div.wrapper > article.post > header.post-header > h2.post-title`.

Drzewko dostępności – Firefox

Nagłówek

Nagłówek



Turn Off Accessibility Features | [beta](#) Check for issues: None

Role	Name
document:	"Document"
heading:	"Nagłówek"
text leaf:	"Nagłówek"
text container:	" "
text leaf:	"Nagłówek"

ARIA

- To standard ułatwiający tworzenie dostępnych aplikacji internetowych.
- Jest bardziej techniczny od WCAG.
- Specyfikacja: <https://w3c.github.io/aria/>
- Dobre praktyki: <https://w3c.github.io/aria-practices/>
- Jeszcze więcej dobrych praktyk: <https://w3c.github.io/using-aria/>

Obsługa klawiaturą

- Określona we WCAG 2.1 w 2.1.1/2.1.3 (Klawiatura).
- Wszystkie funkcje na stronie są możliwe do obsługi z poziomu samej klawiatury.
- Konwencja nakazuje, żeby przyciski aktywowały się po naciśnięciu spacji i Entera.
- Obsługa klawiatury wymaga obsługi fokusu – a więc atrybutu `[tabindex]` →
<https://bitsofco.de/how-and-when-to-use-the-tabindex-attribute/>

Focus i inne stany

- Określone przez WCAG 2.1 w 2.4.7 (Widoczny fokus) oraz w 1.4.11 (Kontrast elementów nietekstowych).
- Przycisk w różnych stanach powinien mieć kontrast 3:1 względem swojego otoczenia.
- Sfocusowany element musi mieć widoczny wskaźnik focusu.
- O innych stanach WCAG nie wspomina, ale dla poprawienia UX warto je dodać → <https://accessibleweb.com/question-answer/what-are-the-contrast-requirements-for-an-elements-focus-mouseover-select-states/>
- W przyszłości, we WCAG 2.2, pojawi się wymóg posiadania kontrastu 3:1/4.5:1 między początkowym stanem a focusem (2.4.11/2.4.12).
- Więcej info: <https://www.sarasoueidan.com/blog/focus-indicators/>



CKSOURCE™

CKSOURCE™

CKSOURCE™

CKSOURCE™

Tryb wysokiego kontrastu w Windowsie

CKSOURCE™

CKSOURCE™

CKSOURCE™

CKSOURCE™

Example

Press me

Press me

Press me

Natywny przycisk działa lepiej

Natywny przycisk

- Problemy ze stylowaniem już nie istnieją → <https://github.com/necolas/react-native-web/issues/1899>
- Obsługę klawiatury i focus dostajemy w gratisie.
- Drzewko dostępności z automatu wie, o jakim elemencie mówimy.
- Tryb wysokiego kontrastu również działa bez problemu:

Click me

Use the Platform, Luke!

Zatem mamy dostępny przycisk?

Large button

Large button

Large button

Large button



Well yes, but actually no

JRCE

JRCE™

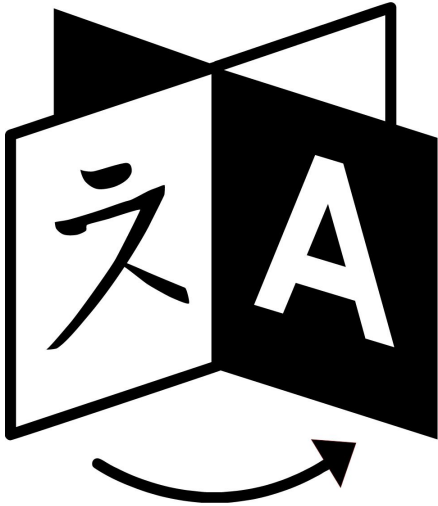
CKSC

CKSC

CKSC



Etykieta tekstowa



Language Icon:

<http://www.languageicon.org/>

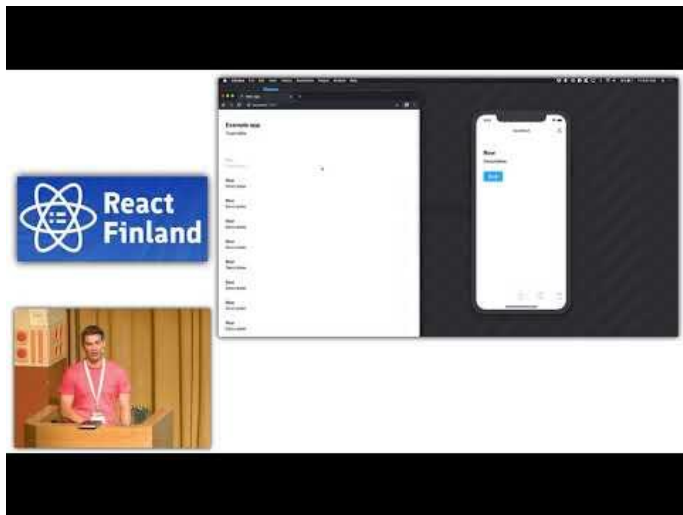
Łapka czy zwykły kursor?

- Łapka oznaczała pierwotnie linki.
- Wiele design systemów rezerwuje łapki właśnie dla linków.
- Domyślne style w systemie i przeglądarce nie dodają łapki do przycisków.
- Więcej info: <https://adamsilver.io/blog/buttons-shouldnt-have-a-hand-cursor/>

ALE

- Używanie łapki do przycisków jest mocno popularną konwencją.
- Przykład: Bootstrap →
<https://getbootstrap.com/docs/5.1/components/buttons/>

A co z urządzeniami dotykowymi?



Rick Hanlon, The Untouchable Web,
<https://www.youtube.com/watch?v=LhKgIxQT4sU>

Przyciski są trudne – dowód anegdotyczny

Adobe wypuściło *serię* artykułów o tworzeniu przycisku:

- <https://react-spectrum.adobe.com/blog/building-a-button-part-1.html>
- <https://react-spectrum.adobe.com/blog/building-a-button-part-2.html>
- <https://react-spectrum.adobe.com/blog/building-a-button-part-3.html>

Rezultatem jest zwykły przycisk →

<https://react-spectrum.adobe.com/react-spectrum/Button.html>



Save

Inne nieoczywiste problemy

Animacje

- Określone we WCAG 2.1 2.3 (Ataki padaczki).
- Intensywne błyskanie może powodować napady padaczkowe.
- Użytkownicy mogą życzyć sobie, by animacje na stronie były wyłączone.
- O preferencji użytkownika dowiemy się z media query `prefers-reduced-motion`.
- Więcej info: <https://www.webkrytyk.pl/2021/08/31/wpadki-i-wypadki-13/>

Gładki zwój

- Płynne przewijanie to też animacja!
- Przewinięcie powinno przerzucić focus do nowej sekcji.
- Natywny `scroll-behavior: smooth;` ma swoje problemy.
- Więcej info:
 - <https://css-tricks.com/smooth-scrolling-accessibility/>
 - <https://schepp.dev/posts/smooth-scrolling-and-page-search/>
 - <https://www.webkrytyk.pl/2020/09/30/wpadki-i-wypadki-12/>

Hamburger czy kebab

- Brak etykiety tekstowej może utrudnić rozpoznanie przycisku.
- Umieszczenie przycisku jest istotne.
- Wypada odpowiednio sterować focusem.
- Animacja przycisku jest animacją.
- Więcej info:
 - <https://inclusive-components.design/menus-menu-buttons/#navigationmenubuttons>
 - <https://www.webkrytyk.pl/2020/01/31/wpadki-i-wypadki-10/>

Combobox

- Wszyscy go tworzą, nikt nie wie do końca jak.
- Sama definicja zmieniała się w różnych wersjach standardu ARIA.
- Ostatecznie combobox to `select` z opcjonalną możliwością wpisywania wartości.
- Więcej info: <https://w3c.github.io/aria-practices/#combobox>

Jak żyć?

Accessibility First Design

- Zawsze bierzmy pod uwagę preferencje użytkownika:
 - Część podsunie nam przeglądarka (np. `prefers-reduced-motion`).
 - O części użytkownik może zdecydować sam (np. przełączanie między jasnym a ciemnym motywem).
- Projektujmy tak, aby WCAG było podstawą designu.
- Myślmy w kategoriach samolubnej dostępności:
 - <https://adrianroselli.com/2018/09/selfish-accessibility-at-codedaze.html>
 - <https://blog.comandeer.pl/to-tylko-niepelnosprawni.html>
- Podchodźmy inkluzywnie: <https://inclusivedesignprinciples.org/>
- I najważniejsze – testujmy!

Testowanie

- Rozwiązania zautomatyzowane (axe, WebAIM, Lighthouse, pa11y...)
- Własnoręcznie
- Metoda korytarzowa
- Audyt przez profesjonalistę
- Testy z użytkownikami

I to by było na tyle!

Czy są jakieś pytania?