Co każdy webmaster o dostępności wiedzieć powinien

Coraz częściej mówi się o dostępności witryn internetowych. Niestety większość deweloperów nadal traktuje ten temat pobieżnie, a część z nich w ogóle nie wie, czym ona jest. Co oznacza pojęcie „dostępna strona" i dlaczego warto się tym tematem zainteresować?

CO, JAK, PO CO I DLA KOGO?

Dostępność strony internetowej zakłada przede wszystkim to, że będzie ona dostosowana dla wszystkich użytkowników sieci. Szczególny nacisk kładzie się na jej przystosowanie do osób o wszelkich niepełnosprawnościach i schorzeniach – zarówno fizycznych, jak i psychicznych. Należy mieć na uwadze istnienie takich osób oraz ich potrzebę dostępu do informacji. Może się przecież zdarzyć, że po drugiej stronie znajduje się osoba, która nie widzi, bądź straciła rękę w wyniku wypadku, przez co nie może posługiwać się myszką. Według szacunków Międzynarodowej Organizacji Zdrowia na całym świecie żyje ponad miliard osób dotkniętych niepełnosprawnością, co stanowi ok. 15% populacji (dane za rok 2016). Należy pamiętać, że kiedy nadarzy nam się okazja tworzenia strony dla instytucji publicznej, mamy prawny obowiązek jej dostosowania do potrzeb osób niepełnosprawnych (Rozporządzenie Rady Ministrów w sprawie Krajowych Ram Interoperacyjności – Dz.U. 2016 poz. 113). W celu zunifikowania wymagań stawianych wobec dostępnych witryn internetowych powstał dokument WCAG (Web Content Accessibility Guidelines) (obecnie wersja 2.0), który zawiera wskazówki dotyczące budowy dostępnych serwisów internetowych.

Najczęstsze błędy deweloperów

1. Niestosowanie tekstów alternatywnych (atrybut „alt") lub ich zły dobór (np.: `<img src="../images/logo.png" alt="logo">` zamiast `<img src="../images/logo.png" alt="Nazwa strony – strona główna">`);
2. Zbyt niski kontrast elementów na stronie;
3. Niewłaściwe wykorzystywanie elementów semantycznych HTML5;
4. Niewłaściwe posługiwanie się nagłówkami;
5. Stosowanie zbyt dużej liczby animacji.

PROJEKTOWANIE STRONY

Strona już na etapie projektowania powinna się charakteryzować odpowiednim podejściem do tematu dostępności. Przede wszystkim należy pamiętać o niestosowaniu zbyt dużej ilości animacji. Choć są one nieodzowną częścią współczesnego web designu, wypada pamiętać o zdrowym umiarze, gdyż mogą one negatywnie wpływać na osoby dotknięte epilepsją bądź nerwicą. Kolejną bolączką stron, już na etapie tworzenia layoutu, jest zbyt niski kontrast elementów, przez co osoby niedowidzące mogą mieć spory problem z przeczytaniem opublikowanych przez nas treści. Pojęcie „odpowiedniego kontrastu" jest jednak względne. To, co dla dobrze widzącej osoby jest „wystarczające", może sprawiać problemy osobom z wadą wzroku. Według wspomnianej normy WCAG 2.0 za minimalny uznaje się kontrast na poziomie 4,5 do 1, a zalecany to 7 do 1. Zalecenie to dotyczy nie tylko tekstu, ale także obrazków zawierających tekst. Tutaj może nasuwać się wniosek, że im wyższy kontrast, tym lepiej. Otóż nie do końca. Badania pokazują, że dla osób z dysleksją łatwiejsze jest czytanie tekstu na tle o niezbyt wysokim poziomie kontrastu. W związku z tym nie zaleca się stosowania jaskrawych kombinacji (m.in. czarny + biały lub czarny + żółty). Do pomiaru kontrastu możemy wykorzystać szereg stworzonych do tego celu programów (m.in. Colour Contrast Analyser lub narzędzia dostępnego na stronie webaim.org).

Niemal na każdej stronie widzimy zdjęcia – czy to w galerii, czy na podstronach w celu przedstawienia produktu lub usługi. Dobre zdjęcie niejednokrotnie potrafi przekazać więcej niż słowa! Ważne więc jest ich odpowiednie oznaczenie. Służy temu atrybut `alt`. Zdecydowanie zbyt często traktowany pobieżnie – niestosowany w ogóle lub źle dobrany. Wartość tego atrybutu powinna jednoznacznie mówić o tym, co przedstawia dane zdjęcie.

Mimo że standardem we współczesnym web dewelopmencie jest stosowanie języka JavaScript, należy mieć na uwadze, aby w przypadku, gdy przeglądarka użytkownika z jakichś powodów nie obsługuje tego języka, nasza strona nadal potrafiła dostarczyć odbiorcy treść. Sytuacji, w których skrypt JS nie zostanie uruchomiony, może być kilka (np. błąd w nowej wersji przeglądarki lub zerwanie połączenia z serwerem w trakcie dosyłania użytkownikowi plików JS). Słowem: skrypt JS nie może blokować ani uniemożliwiać wyświetlania treści. Takie podejście nosi nazwę Progressive Enhancement – zakłada całkowitą niezależność warstw (treści, metadanych, wyglądu oraz mechaniki działania) oraz ich właściwą komunikację. Treść, jaką chcemy przekazać, musi bezwzględnie trafić do osób chcących ją otrzymać! Pamiętajmy, że treść pozbawiona „fajerwerków" tworzonych poprzez CSS, JS itd. nadal jest wartościowa (a jeśli nie jest, to czas najwyższy coś z tym zrobić).

Współczesne strony często zawierają dynamiczne sekcje, których zawartość zmienia się po wykonaniu określonej akcji przez użytkownika. Przykładem takiego rozwiązania są m.in. zakładki czy też akordeony. Język HTML nie przewiduje na tę okoliczność specjalnych znaczników, dlatego też wyżej wspomniane elementy budowane są ze znaczników niesemantycznych takich jak `<div></div>`, czy `<span></span>`. Dla osób niewidomych bądź niedowidzących stanowią one problem, tym bardziej, że ich działanie nie jest wspierane przez technologie asystujące. Aby nadrobić te niedociągnięcia i naprowadzić czytniki na właściwy trop stosuje się atrybuty `ARIA`. Dzięki nim możemy dodać semantyczne znaczenie dowolnym elementom stronie (w tym również bardziej zaawansowanym dodatkom).

Podstawowe atrybuty ARIA

1. Role (`np. role="tab"`) – przeznaczone do opisywania przeznaczenia danego elementu
» `alert`
» `slider`
» `tooltip`
» `tab`
» `toolbar`
» `banner`
» `search`
2. Stany i właściwości
» `aria-hidden`
» `aria-label`
» `aria-checked`
» `aria-readonly`
» `aria-required`
» `aria-atomic`
» `aria-busy`
» `aria-dropeffect`
» `aria-grabbed`
» `aria-controls`

Jeśli w jakimś miejscu na naszej stronie treść jest generowana dynamicznie poprzez skrypty języka JavaScript, warto rozważyć zastosowanie znacznika `<noscript></noscript>`, który w przypadku niezaładowania skryptów pokaże użytkownikowi domyślną, wcześniej ustaloną treść. Takie rozwiązanie sprawi, że nie zostawimy użytkownika „z niczym".

Na tym etapie powinniśmy się też zastanowić nad rozmieszczeniem tekstu, odpowiednim zredagowaniu tytułów oraz podzieleniu treści na odpowiednie sekcje. Treść główna strony powinna posiadać trafne tytuły, które pomogą użytkownikowi w zorientowaniu się w jej treści. Tekst powinien zostać podzielony na tematyczne sekcje (`<section></section>`) opatrzone nagłówkami (`<header></header>`). Wspomniane tytuły powinny posiadać odpowiednie oznaczenie (h1-h6), które określa ich hierarchię. Zaleca się także, aby sam tekst był odpowiednio sformatowany. Mile widziane jest justowanie tekstu jedynie do strony lewej, gdyż jego rozciągnięcie może wywoływać wrażenie „ściany tekstu" uniemożliwiającej jego przeczytanie osobom z dysleksją. Jeśli w tekst wplatamy linki, należy je odpowiednio wyróżnić, aby przeglądający mógł je bez trudu zauważyć. W przypadku gdy na naszej stronie udostępniamy materiały audio lub wideo, powinniśmy udostępnić również alternatywne wersje dla osób niesłyszących bądź niewidomych (np. transkrypt filmu).

DOSTĘPNE MENU

Jednym z kluczowych elementów na każdej stronie jest menu. Zarówno osobom zdrowym, jak i niepełnosprawnym pomaga w odnalezieniu się na stronie. Jednak niewłaściwie zaprojektowane lub wdrożone może powodować więcej problemów niż jego brak. W swoich założeniach dostępne menu powinno być intuicyjne w obsłudze, a jego znalezienie nie powinno sprawiać nikomu najmniejszego problemu.

Zacznijmy od tego, że menu powinno być zbudowane z wykorzystaniem elementów semantycznych HTML5. Mowa tutaj o znaczniku `<nav></nav>`.

Kolejną ważną kwestią jest umożliwienie użytkownikowi korzystanie z elementów menu, nawet gdy ten nie ma możliwości korzystania z myszki. Osoby dotknięte schorzeniami ruchowymi takimi jak np. choroba Parkinsona często zmuszone są korzystać z alternatywnych urządzeń wejścia, takich jak head wand, mouth stick czy też single switch access oraz oczywiście klawiatura (zwykła bądź brajlowska).

Wszystkie te urządzenia mimo zdecydowanie różniącej się budowy opierają się na podobnej zasadzie działania.

Najprostszą metodą stworzenia dostępnego menu jest korzystanie z semantycznych znaczników HTML5, takich jak `<a href="#">Odnośnik</a>` lub `<button>Przycisk</button>`. Te elementy z uwagi na swoje semantyczne znaczenie od razu umożliwiają dostęp z poziomu klawiatury. Niestety na niektórych stronach nadal panuje semantyczny bałagan, przez co osoby posługujące się klawiaturą nie mają łatwego zadania, aby dostać się w wybrane przez siebie miejsce na stronie. W Internecie znajdziemy też „kwiatki", które w nazbyt jasny sposób mogą posłużyć jako antywzorzec. Przyjrzyjmy się jednemu z nich:

Listing 1. Przykład niewłaściwego wykonania menu

```
<div id="menu">
<ul>
<li onclick="window.location.href='http://niesemantycznastrona.
pl'">Główna</li>
<li onclick="window.location.href='http://niesemantycznastrona.
pl/o-nas'">O nas></li>
<li onclick="window.location.href='http://niesemantycznastrona.
pl/kontakt'">Kontakt></li>
</ul>
</div>
```

Jak widać powyżej, nie został zastosowany ani jeden element semantyczny, który pomógłby (nawet w najmniejszym stopniu)

w nawigacji. Ponadto w sytuacji, gdy użytkownik ma wyłączoną obsługę skryptów w przeglądarce, elementy listy stają się kompletnie bezużyteczne (już pomijając wykorzystanie archaicznej metody przypisania zdarzenia – mowa tutaj o „onclick").

Aby powyższe menu stało się semantyczne, wystarczyłoby do każdego elementu listy dodać odnośnik (np.: `<a href="http://niesemantycznastrona.pl">Główna</a>`), a `div` od `id="menu"` zastąpić przez znacznik `<nav></nav>`.

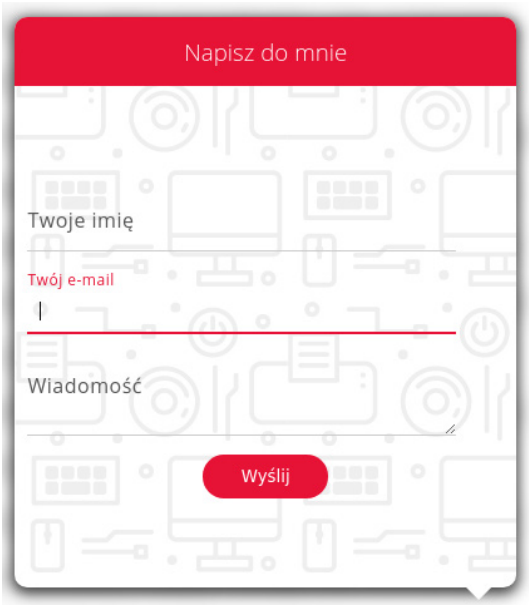
Niemal każde menu może poszczycić się przeróżnymi animacjami, które sygnalizują użytkownikowi najechanie kursorem na dany jego element. Warto także te animacje przystosować do obsługi poprzez klawiaturę. W tym celu oprócz standardowego selektora `:hover` zastosować można również `:focus`. Wówczas po wskazaniu danego elementu za pomocą klawiatury (przycisk Tab) zostanie pokazany identyczny efekt jak wówczas, gdy najedziemy na niego kursorem myszy.

Coraz popularniejszym trendem staje się stosowanie ukrytego menu, dostępnego po kliknięciu odpowiedniego przycisku (tzw. hamburgera). Pamiętajmy, aby ten element strony również był prawidłowo zbudowany. Po pierwsze, taki przycisk posiadał własną, odpowiadającą mu etykietę. Można to osiągnąć poprzez wpisanie jej wewnątrz znacznika button (`<button>Menu</button>`). W przypadku gdy jest to niemożliwe (gdy np. przycisk jest tylko grafiką i nie ma w nim miejsca na tekst), powinniśmy zastosować atrybut `aria-label` (`<button aria-label="Otwórz menu"></button>`). Podobnie jak w przypadku elementów menu również przycisk jego otwarcia powinien otrzymywać widoczny focus klawiatury. Jeśli zastosowaliśmy poprawny element semantyczny (`<button></button>`), nie musimy się martwić, gdyż otrzymuje on focus nawet bez naszej ingerencji. Sam focus powinien być na tyle widoczny, aby po jego wybraniu osoba korzystająca z naszej strony nie miała wątpliwości co do jego stanu. Dobrą praktyką jest ułożenie w kodzie menu bezpośrednio pod przyciskiem otwierającym. W ten sposób po kliknięciu w przycisk focus zostanie automatycznie przeniesiony na pierwszy element menu. Na koniec warto poinformować użytkownika o stanie przycisku za pomocą atrybutu `aria-expanded`. Za jego pomocą informujemy użytkownika o tym, czy menu jest otwarte, czy zamknięte (`<button aria-expanded="false" type="button"></button>`). Wartość tego atrybutu powinna być zatem kontrolowana przez skrypt języka JavaScript.

DOSTĘPNE FORMULARZE

Formularze na stronie są niezwykle istotne z punktu widzenia utrzymywania kontaktu z Klientem. Ich odpowiednie zaprojektowanie oraz prawidłowa implementacja umożliwia zachowanie prawidłowego przebiegu komunikacji również z osobami dotkniętymi niepełnosprawnością.

Jedną z najczęściej pomijanych kwestii dotyczących formularzy są odpowiednie oznaczenia pól oraz przycisków w nich używanych. W większości współczesnych stron na potęgę wykorzystywany jest atrybut `placeholder`. Jest on o tyle nieprzyjazny dla odwiedzających, że zazwyczaj występuje sam (bez odpowiadającemu polu znacznika `label`). Powoduje to, że gdy uaktywnimy pole zawierające jedynie ten atrybut, możemy zwyczajnie zapomnieć, co mieliśmy w to pole wpisać. Poprawnie oznaczone pole powinno zawierać odpowiadającą mu etykietę (`<label for="">Treść etykiety</label>`, gdzie `for=""` ma zawierać nazwę odpowiadającego mu pola). Co ważne: taki zabieg wcale nie musi oszpecić naszej strony. Wręcz przeciwnie – istnieje wiele pomysłów na kreatywne wykorzystanie znacznika `<label></label>`.



Rysunek 1. Przykład zastosowania znacznika <label></label>

Podobnie jak w przypadku menu każdy element wykorzystany w formularzu kontaktowym powinien zawierać widoczny dla użytkownika focus, a także być możliwym do obsłużenia z poziomu klawiatury bądź innego alternatywnego urządzenia wskazującego.

Przydatne narzędzia do sprawdzania dostępności stron internetowych

1. www.wave.webaim.org – walidator umożliwiający łatwe przeanializowanie pojedynczej strony pod kątem dostępności według standardu WCAG 2.0;
2. Colour Contrast Analyser – program badający kontrast strony internetowej (program dostępny na platformy Windows i MacOS);
3. PDF Accessibility Checker – narzędzie pozwalające sprawdzić dostępność plików PDF;

BARTŁOMIEJ KRAKOWSKI
bartlomiej.krakowski@interia.eu

Web developer i web designer. Zwolennik czystego kodu i wolnego oprogramowania. Osoba zafascynowana możliwościami, które daje nam Internet, a także technologią, która nas otacza. Obcowanie z internetowymi witrynami przestało mu wystarczać, więc zaczął tworzyć je sam.