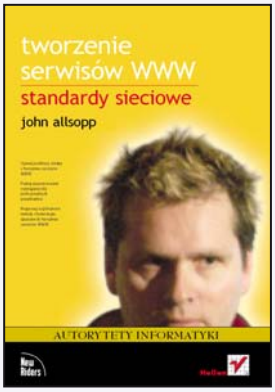
Tworzenie serwisów WWW. Standardy sieciowe



Autor: John Allsopp
Tłumaczenie: Dagmara Suma
ISBN: 978-83-246-2657-1
Tytuł oryginału: Developing with Web Standards
Format: 158×235, stron: 480

Krajobraz nowoczesnych technologii WWW

- Opanuj podstawy wiedzy o tworzeniu serwisów WWW
- Poznaj zaawansowane rozwiązania dla profesjonalnych projektantów
- Rozpracuj współczesne metody i technologie, używane do tworzenia serwisów WWW

Z pewnością znasz książkę Projektowanie serwisów WWW. Standardy sieciowe Jeffreya Zeldmana. Jeśli jesteś zagorzałym fanem tego kultowego podręcznika, niewątpliwie zainteresuje Cię również ten tom towarzyszący. Jego autor – instruktor oraz twórca licznych propozycji standardów sieciowych – oferuje wiedzę na temat architektury stojącej za profesjonalnymi witrynami WWW, wskazując przy tym sposoby stosowania standardów z czysto taktycznego i praktycznego punktu widzenia.

Książka ta jest przeznaczona dla żółtodziobów chcących nauczyć się budować strony WWW, a także dla doświadczonych twórców, szukających głębszej wiedzy, pomagającej wyjść poza metody i modele, które już dobrze znają. Zawiera mnóstwo sztuczek i chwytów, choć jej podstawowym celem jest pomoc w budowaniu systematycznej wiedzy na temat praktyki tworzenia serwisów WWW.

Poznaj:

- podstawowe technologie tworzenia front-endu – HTML i XHTML, CSS i DOM;
- zalecane metody tworzenia łatwych w użyciu serwisów WWW;
- praktykę tworzenia serwisów WWW: semantyczny układ znaczników, techniki radzenia sobie z niezgodnościami i błędami przeglądarek, układy stron oparte na CSS oraz tematykę platform CSS;
- technologie wchodzące na rynek: przełomowe narzędzia, z których możesz już zacząć korzystać, w tym HTML5, CSS3, czcionki WWW, SVG i Canvas.

Witryna WWW związana z oryginalnym wydaniem książki Tworzenie serwisów WWW. Standardy sieciowe (Developing with Web Standards) jest dostępna pod adresem devwws.com. Znajdziesz tam więcej informacji i dodatków, będziesz też mógł osobiście skontaktować się z autorem książki.

John Allsopp jest współzałożycielem witryny westciv.com oraz twórcą programu Style Master, cieszącego się wielką sławą narzędzia do tworzenia arkuszy stylów na różnych platformach. Jest również autorem wielu kursów, materiałów szkoleniowych, narzędzi, instrukcji oraz artykułów przeznaczonych dla projektantów czy twórców serwisów WWW, w tym bardzo ważnego artykułu The Dao of Web Design, opublikowanego na witrynie A List Apart. Pełni też funkcję współzarządcy nowej, działającej w ramach organizacji W3C, Incubator Group, której prace koncentrują się na kształceniu kolejnych pokoleń profesjonalnych twórców WWW (www.w3.org/2005/Incubator/owea).

Spis treści

Część I > Podstawy

Część II > Sztuka tworzenia w świecie rzeczywistym

ROZDZIAŁ SZÓSTY

Dostępność

Gdy Tim Berners-Lee tworzył zręby World Wide Web we wczesnych latach 90. ubiegłego wieku, szczególnie mocno akcentował znaczenie jej powszechności i uniwersalności. Dostępność dla wszystkich ludzi, niezależnie od ich ewentualnej niepełnosprawności, była fundamentalną częścią tej wizji.

Przez ponad dziesięć lat organizacja W3C opracowywała protokoły, których celem było sprawienie, aby sieć WWW była jak najbardziej dostępna, i włączała czynniki odpowiedzialne za zapewnianie dostępności do specyfikacji regulujących standardy takie jak HTML czy CSS. Równolegle z tym ciała ustawodawcze na całym świecie wprowadzały przepisy prawne związane z dostępnością oraz regulacje, które miały taki sam wpływ na sieć WWW jak na środowisko fizyczne i które często bezpośrednio odwoływały się do tych specyfikacji W3C.

Mimo to twórcy serwisów WWW wydają się nieraz bagatelizować kwestie związane z dostępnością. W najlepszym przypadku są one rozważane na samym końcu, stanowią listę formalności, które muszą zostać dopełnione jak najmniejszym wysiłkiem i kosztem finansowym już po całkowitym zakończeniu „prawdziwej" pracy nad projektem.

Jednak zaangażowanie w zapewnianie dostępności powinno być jedną z podstaw etyki naszej rodzącej się profesji: czymś, co z pełną świadomością powinniśmy starać się osiągnąć, nie zaś rzeczą, którą robimy niechętnie i tylko wtedy, gdy wymaga tego od nas prawo. Jeśli na co dzień utrzymujesz kontakty z osobą niepełnosprawną, z pewnością wiesz, jak trudne może być uzyskanie dostępu do istotnych informacji, sieci społecznych i usług. Jeśli nie masz tej okazji, spróbuj wyobrazić sobie ograniczenie swoich własnych możliwości dostępu i zastanowić się, w jakich sytuacjach znajdują się często ludzie niepełnosprawni.

Przez tydzień rób zakupy przez sieć i tylko w ten sposób utrzymuj swoje kontakty biznesowe. Korzystaj z czytnika ekranowego w celu sprawdzania swojej poczty elektronicznej i najświeższych informacji ze świata. Nawiguj, używając wyłącznie klawiatury lub jedynie myszy. Szybko nabierzesz szacunku dla ogromnych wyzwań, które większość serwisów WWW i usług internetowych stawia przed ludźmi niepełnosprawnymi, wyzwań, które masz okazję zmniejszyć z racji wykonywanego zawodu.

Kluczowa rola odgrywana przez sieć WWW w poprawianiu standardu życia tak wielu osób niepełnosprawnych jest jednym z powodów, dla których zostałem twórcą serwisów. Nie musi tak być w przypadku każdego, niezależnie jednak od tego, czy Twoją motywacją jest altruizm, czy też wymogi prawne i groźba wytoczenia procesów sądowych, o których głośno bywa ostatnimi czasy w Stanach Zjednoczonych i Australii, przekonasz się, że zapewnienie należytej dostępności jest jednym z podstawowych zadań związanych z profesjonalnym tworzeniem serwisów WWW.

Na szczęście, jest to też znacznie mniej kłopotliwe i wcale nie tak trudne do osiągnięcia, jak nauczono Cię wierzyć.

W rozdziale tym nie będziemy w stanie bardzo szczegółowo opisać wszystkich zagadnień związanych z zapewnianiem dostępności, ale postaramy się zająć najważniejszymi kwestiami, zaleceniami oraz wyzwaniami technicznymi dotyczącymi tej materii, a także przyjrzeć kilku prostym krokom, które pomogą nam tworzyć odpowiednio dostępne serwisy WWW przy niewielkim tylko dodatkowym wysiłku z naszej strony.

Biznesowe argumenty za zapewnianiem dostępności

Za koniecznością zapewniania dostępności serwisów WWW przemawiają także solidne argumenty natury biznesowej. Tworzenie takich serwisów może wpłynąć na poprawienie komfortu życia odwiedzających je osób, przekłada się też jednak na bezpośrednie oszczędności (związane na przykład ze zmniejszeniem liczby użytkowników dzwoniących na numery telefonicznej obsługi klienta) i umożliwia biznesowi dotarcie do znacznie większej grupy ludzi, co wpływa z kolei na wzrost sprzedaży i zysków. Liczba osób niepełnosprawnych w naszych społecznościach jest o wiele większa, niż gros ludzi sobie wyobraża. Wydział Zdrowia i Pomocy Humanitarnej Stanów Zjednoczonych ogłosił na przykład niedawno, że jeden na pięciu obywateli USA cierpi z powodu jakiegoś rodzaju niepełnosprawności, a w przypadku jednej osoby na osiem niepełnosprawność ta ma charakter poważny.

Gdy mówimy o osobach niepełnosprawnych, mamy na myśli dziesiątki milionów ludzi, a także segment rynku, który znacznie częściej korzysta z handlu elektronicznego niż jakakolwiek inna grupa społeczna. Zaspokajanie potrzeb tej wspólnoty może bezpośrednio wpłynąć na rentowność niejednej firmy.

Oprócz tych zysków o charakterze czysto komercyjnym zastosowanie technik tworzenia dostępnych serwisów WWW może mieć również szereg innych pozytywnych skutków. Google i inne mechanizmy wyszukiwania są w pewnym sensie ślepe. Jedynym tekstem, jaki widzą, gdy indeksują serwis, jest prawdziwy *tekst*, nie zaś wszelkie obrazy czy pliki Flash, które mogą zostać właściwie zinterpretowane tylko przez ludzi dysponujących odpowiednio dobrym wzrokiem. Serwisy WWW, w przypadku których zastosowano dobrą i przejrzystą strukturę semantyczną, uniknięto używania kodu opartego na tabelach oraz zadbano o wykorzystanie innych technik zapewniania dostępności, mogą dzięki temu cieszyć się lepszą „wykrywalnością" i pozycjonowaniem przez mechanizmy wyszukiwania.

Prawodawstwo na świecie

Ostatnimi czasy na całym świecie uchwala się akty prawne mające zwalczać dyskryminację osób niepełnosprawnych. Przepisy tego rodzaju dotyczą zwykle wielu różnych aspektów życia, coraz większy jednak nacisk kładzie się tu również na kwestie związane z dostępnością sieci i serwisów WWW. Problem ten jest bardzo złożony, dlatego wszelkie próby prostego zaprezentowania go tutaj skazane są na niepowodzenie i mogą wprowadzić tylko jeszcze większe zamieszanie. Źródła, z których możesz czerpać bardziej obszerną wiedzę na temat sytuacji na świecie w tej materii, zostały wymienione na końcu niniejszej książki.

Dostępność i W3C

Od pierwszych dni swojego istnienia organizacja W3C podejmuje próby tworzenia zaleceń, wytycznych i wskazówek dotyczących kwestii zapewniania dostępności, a także wbudowania ich bezpośrednio w standardy takie jak CSS i HTML. W roku 1999 W3C opublikowała pierwszą wersję zbioru dokumentów znanych jako „Wytyczne dotyczące dostępności treści internetowych" (ang. *Web Content Accessibility Guidelines*, w skrócie — WCAG 1), zaś w roku 2008 udostępniła jego zaktualizowaną wersję (znaną jako WCAG 2). Dodatkowo, aby zaspokoić rosnącą potrzebę wytycznych związanych z dostępnością aplikacji WWW, działająca w obrębie W3C inicjatywa

dostępności do sieci (ang. *Web Accessibility Initiative*, w skrócie — WAI) opracowała regulujący te kwestie standard Accessible Rich Internet Applications Suite (WAI-ARIA), którym zajmiemy się już niebawem.

WCAG 1

Opublikowany w 1999 roku, w czasach panowania przeglądarek wersji czwartej, WCAG 1 był pierwszą ważną próbą skodyfikowania zbioru procedur mających zapewnić większą dostępność serwisów WWW. Jego zadaniem było pokazanie sposobu „w jaki można tworzyć treści internetowe dostępne dla osób niepełnosprawnych" (jak można przeczytać już w pierwszym zdaniu opisu WCAG 1 znajdującego się pod adresem: *www.w3.org/TR/WCAG10*).

WCAG 1 został podzielony na czternaście wytycznych, z których każda dzieli się na punkty kontrolne, a te z kolei mają priorytety od A do AAA (określane często jako „potrójne A"). Zgodność z wytycznymi WCAG 1 może być mierzona stopniem zgodności z tymi punktami kontrolnymi. Dokument ma zgodność poziomu A, jeśli zapewnia zgodność ze wszystkim punktami kontrolnymi poziomu A. Analogicznie, dokument uważany jest za zgodny z poziomem AA („podwójne A"), jeśli spełnia wymogi wszystkich punktów kontrolnych AA, zaś z poziomem AAA, gdy zapewnia zgodność z wszystkimi punktami kontrolnymi AAA.

Zanim bardziej zagłębimy się w tematykę tych wytycznych i punktów kontrolnych, należy tu odnotować, że zgodność AAA uważa się ogólnie za trudną do osiągnięcia i nie zawsze jest warta włożonego w nią wysiłku, ponieważ wiele punktów kontrolnych AAA ma bardzo subiektywny charakter, a pełna zgodność z poziomem AAA ma dość niewielkie znaczenie praktyczne. W związku z tym przez większość ekspertów (a także niektóre oficjalne wytyczne rządowe) zgodność poziomu AA jest zalecana jako wystarczająca.

Wytyczne i punkty kontrolne

Jak już wspomnieliśmy powyżej, każda z czternastu wytycznych WCAG 1 zawiera kilkanaście punktów kontrolnych. W rozdziale tym przyjrzymy się *tylko tym punktom kontrolnym*, które najczęściej sprawiają trudności. Wytyczne WCAG 1 (oraz towarzyszący im dokument „Techniki") są dość proste, nie bój się więc przeczytać ich samodzielnie.

Wytyczna 1: Zapewnij równoważne odpowiedniki treści dźwiękowych i wizualnych

Wytyczna ta ma na celu zaspokojenie potrzeb osób mających problemy ze słuchem lub wzrokiem. Jednym z najbardziej powszechnych błędów popełnianych przez twórców serwisów WWW jest pomijanie treści **alt** związanych z obrazami. W kodzie HTML 4 wszystkie elementy **img** muszą mieć atrybut **alt**, choć wartość tego atrybutu może być pusta (a więc dopuszczalne jest tu wyrażenie **alt=""**) w przypadku grafik pełniących funkcje wyłącznie dekoracyjne. W takich przypadkach zdecydowanie powinniśmy używać możliwości oferowanych przez CSS, zamiast dodawać tego typu obrazy za pomocą kodu HTML.

Zapewnianie dostępności treści audio i wideo wymaga znacznie więcej niż tylko prostych odpowiedników tekstowych, a opis niezbędnych technik, które się tu wykorzystuje, znacznie wykracza poza zakres materiału prezentowanego w tej książce. Łącza do źródeł informacji na ten temat oraz wiadomości dotyczących sposobów zapewniania dostępności treści audio i wideo znajdziesz na końcu niniejszej publikacji.

Wytyczna 2: Nie polegaj wyłącznie na kolorze

Zalecenie to ma zastosowanie raczej do kodu CSS niż HTML, ponieważ formatowanie zawsze powinno być definiowane za pośrednictwem mechanizmu CSS, nie zaś prezentacyjnego kodu HTML. Ocenia się, że nawet osiem procent męskiej populacji ma pewnego rodzaju problemy z rozróżnianiem barw, zaś osoby korzystające z urządzeń monochromatycznych (takich jak czytnik Kindle firmy Amazon) nie są w stanie rozróżniać jakichkolwiek kolorów poza odcieniami szarości. Zamiast więc na przykład używać barwy zielonej w celu wskazania bezpiecznego wyboru i czerwieni do oznaczania zagrożenia, powinieneś stosować kształty, etykiety tekstowe i inne rodzaje wskazówek przekazujących tę informację.

Wytyczna 3: Korzystaj ze znaczników oraz arkuszy stylów i rób to we właściwy sposób

Wytyczna ta sugeruje, że powinniśmy używać technologii WWW nie tylko zgodnie z literą prawa (układy stron wykorzystujące tabele mimo wszystko przejdą walidację), lecz również zgodnie z jego duchem. Należące do niej punkty kontrolne mają między innymi zapewnić, że dokumenty są prawidłowe, w kodzie CSS stosowane są jednostki względne, takie jak **em** i **%**, zamiast **px**, zaś w kodzie HTML używane są odpowiednie elementy semantyczne i strukturalne (na przykład nagłówki, listy, cytaty itd.).

Wytyczna 5: Twórz płynnie przekształcające się tabele

Dane tabelaryczne mogą powodować poważne problemy w przypadku osób korzystających z urządzeń asystujących, takich jak czytniki ekranowe. Wytyczna 5 zawiera kilka punktów kontrolnych, które mają Ci pomóc w poprawieniu dostępności danych tego rodzaju.

Wspomniane punkty kontrolne dotyczą używania nagłówków **`row`** i **`column`** w tabelach danych, zaś w przypadkach skomplikowanych tabel korzystania z takich elementów jak **`thead`**, **`tfoot`** oraz **`tbody`** w celu grupowania wierszy, **`col`** i **`colgroup`** w celu grupowania kolumn, a także atrybutów **`axis`**, **`scope`** i **`headers`** w celu wskazania relacji występujących pomiędzy komórkami i nagłówkami. Sposobami poprawiania dostępności tabel danych zajmiemy się jeszcze w dalszej części niniejszego rozdziału.

Wytyczna 9: Projektuj zgodnie z zasadą niezależności od sprzętu

Wytyczna 9 koncentruje się na znaczeniu możliwości używania serwisu lub strony WWW niezależnie od rodzaju wykorzystywanego urządzenia wejściowego, a więc tego, czy zamiast samej myszy stosowana jest klawiatura, wejściowe urządzenie głosowe czy też sprzęt jeszcze innego typu.

To tylko niektóre z wytycznych, lecz wiele ważnych i popularnych serwisów WWW nie spełnia wymogów stawianych nawet przez nie. Zapewnienie zgodności poziomu AA standardu WCAG 1 jest w dużej mierze sprawą prostą, a stopień tej zgodności da się zwykle ocenić w sposób automatyczny za pomocą odpowiednich narzędzi, którymi zajmiemy się już za chwilę. Oznacza to w praktyce, że dość łatwo można wyznaczyć poziom zgodności serwisów WWW z tymi wytycznymi, a ich twórcy nie dysponują wieloma wymówkami, żeby tego nie robić.

Narzędzia do zapewniania jakości dla WCAG 1

Nie da się sprawdzić mechanicznie zgodności ze wszystkimi wytycznymi WCAG 1, ponieważ do oceny stopnia przestrzegania niektórych z nich wymagany jest udział człowieka, jednak w przypadku tych, dla których jest to możliwe, istnieje sporo narzędzi ułatwiających proces testowania. Należą do nich:

- CynthiaSays firmy HiSoftware — *www.cynthiasays.com*,
- HERA — *www.sidar.org/hera/index.php.en*,
- WAVE firmy Webaim — *wave.webaim.org*,
- Total Validator — *www.totalvalidator.com*,
- ATRC Web Accessibility Checker — *www.achecker.ca/checker/index.php*.

WCAG 2

Standard WCAG 1 powstał z myślą o technologiach WWW istniejących w czasach, gdy został on opublikowany. W kolejnych latach niektóre z nich bardziej dojrzały, zaś inne — takie jak skrypty DOM — zyskały na znaczeniu. Drugie wydanie WCAG, czyli WCAG 2, ma już zatem nieco inną strukturę.

Zasady, wytyczne, kryteria sukcesu i techniki

WCAG 2 powstał w celu utworzenia wytycznych, które są w mniejszym stopniu związane z konkretnymi technologiami, bardziej obiektowe, a — co za tym idzie — znacznie łatwiej poddają się testom (przeprowadzanym za pomocą oprogramowania lub przez ludzi), jak również dają się dostosować do zmian zachodzących w dziedzinie technologii WWW. Zgodnie z oświadczeniem Web Accessibility Initiative WCAG 2 ma znaleźć zastosowanie w przypadku „różnych typów technologii sieciowych i technologii bardziej zaawansowanych", jak również technologii, które „powstaną w przyszłości". Standard został także zaprojektowany w taki sposób, aby zgodność z definiowanymi przezeń kryteriami sukcesu dała się „bardziej precyzyjnie testować za pomocą testów automatycznych oraz analiz prowadzonych przez ludzi" (*www.w3.org/WAI/WCAG20/wcag2faq.html*).

Podczas gdy WCAG 1 był zbiorem wytycznych, z którymi związane były punkty kontrolne, WCAG 2 podzielone zostało na cztery główne zasady. Każda z nich zawiera szereg wytycznych, zaś każda z wytycznych ma pewną liczbę punktów kontrolnych noszących nazwę *kryteriów sukcesu*. Każde z tych kryteriów jest związane z *wystarczającymi* technikami (czyli rekomendowanymi sposobami spełniania tego kryterium sukcesu) oraz technikami *doradczymi* (metodami, które same w sobie nie są wystarczające do spełnienia kryterium, lecz są mimo to zalecane). Żadna z tych technik nie jest *wymagana* w celu spełnienia kryterium. W dokumencie „Techniki WCAG 2" opisano również *typowe defekty*, które są definiowane przez grupę roboczą jako „praktyki twórcze, których zastosowanie sprawia, że treść WWW nie jest zgodna z WCAG 2.0".

WCAG 2 ma swoich krytyków na forum osób zajmujących się kwestiami dostępności, a przyjmowanie go jako podstawowego wyznacznika dostępności treści WWW odbywa się znacznie wolniej, niż organizacja W3C mogła się spodziewać. Wydaje się jednak, że zastąpi on w końcu WCAG 1 jako najważniejsze źródło wytycznych w tej dziedzinie. Przegląd niektórych argumentów przeciwko WCAG 2 wysuwanych przez uczestników w tej dyskusji możesz znaleźć w definicji dostępności treści WWW zamieszczonej w anglojęzycznej

wersji Wikipedii, która w momencie pisania niniejszej książki była zaskakująco dobra (*en.wikipedia.org/wiki/Web_accessibility#Criticism_of_WAI_guidelines*).

WCAG 2 to dokument znacznie większy niż WCAG 1. Został on poddany krytyce głównie za swoją ogólnikowość i zastosowanie niezrozumiałego żargonu — przykład tej krytyki możesz znaleźć w artykułach zamieszczonych pod adresami: *http://www.alistapart.com/articles/tohellwithwcag2* oraz *http://www.webcredible.co.uk/user-friendly-resources/web-accessibility/wcag-guidelines-20.shtml*. Standardowi towarzyszy jednak bardzo dokładna dokumentacja, której zadaniem jest pomaganie twórcom w zrozumieniu i spełnieniu każdego kryterium sukcesu. Opracowano go również w taki sposób, aby strony spełniające w tej chwili wymogi stawiane przez kryteria WCAG 1 A i AA w większości przypadków spełniały też kryteria WCAG 2 bez konieczności wprowadzania żadnych poprawek lub dzięki bardzo niewielkiemu nakładowi pracy.

Szczegółowy opis otchłani i zawiłości standardu WCAG 2 daleko wykracza poza zakres materiału prezentowany w niniejszej książce, jednak wiele interesujących łączy i adresów innych dokumentów związanych z tymi wytycznymi znajdziesz w dodatku „Źródła" zamieszczonym na końcu tej publikacji.

Narzędzia do zapewniania jakości dla WCAG 2

Z uwagi na to, że WCAG 2 nadal znajduje się na stosunkowo wczesnym etapie rozwoju, dostępnych jest też w jego przypadku znacznie mniej zautomatyzowanych narzędzi do zapewniania jakości niż w przypadku WCAG 1. Z zaprezentowanej wcześniej listy rozwiązań dla WCAG 1 jedynie narzędzie ATRC Web Accessibility Checker zapewnia dodatkową możliwość testowania stron pod kątem zgodności z wymogami WCAG 2.

ARIA

Wraz z tym, jak treści i serwisy WWW stawały się bardziej złożone i zaczęły w coraz większym stopniu przypominać aplikacje, wzrastała też trudność zapewniania, a nawet definiowania ich dostępności. W celu rozwiązania tych problemów WAI opracowała pakiet Accessible Rich Internet Applications, zwany też WAI-ARIA lub — częściej — po prostu ARIA. ARIA jest zbiorem rozszerzeń języka HTML, których można używać do opisywania elementów, dzięki czemu możliwe jest identyfikowanie roli, stanów i właściwości każdego z nich w sposób, który znacznie zwiększa dostępność wynikowej strony.

W punkcie tym przyjrzymy się najpierw pobieżnie temu, co umożliwia Ci ARIA. Choć jest ona w dalszym ciągu w początkowej fazie rozwoju, cieszy się naprawdę niezłą obsługą w przypadku wszystkich współczesnych przeglądarek internetowych, w tym także programu Internet Explorer 8, jak również dużej liczby czytników ekranowych, poza tym wydaje się całkiem jasne, że twórcy przeglądarek rzetelnie popierają tę technologię. ARIA pozostanie z nami na stałe.

Dostęp za pomocą klawiatury

Strona czy aplikacja WWW, aby była dostępna, musi być w pełni użyteczna za pomocą urządzeń innych niż mysz. ARIA zapewnia, że aplikacje i strony mogą być używane za pomocą klawiatury, nie wymagając zastosowania myszy.

W standardzie HTML 4 *ognisko wprowadzania* (ang. *focus*) określa, który element strony jako pierwszy otrzymuje dane wejściowe wprowadzone przez użytkownika za pomocą klawiatury lub innego urządzenia wejściowego. Ognisko wprowadzania może przejąć ograniczona liczba elementów, takich jak **`a`**, **`area`**, **`button`**, **`input`** oraz **`select`**, a użytkownik może korzystać z klawisza tabulacji w celu cyklicznego przechodzenia pomiędzy nimi. *Porządek przechodzenia klawiszem tabulacji* (ang. *tab order*) tych elementów (czyli kolejność, w której będą one przejmowały ognisko wprowadzania wraz z tym, jak użytkownik będzie naciskał klawisz tabulacji) jest określany przez kolejność ich występowania w źródłowym kodzie HTML lub też przez wartość atrybutu **`tabindex`**, który za pomocą liczb całkowitych definiuje odpowiedni porządek w następujący sposób: im niższa jest wartość przypisana atrybutowi, tym wcześniejsze miejsce element zajmuje w porządku przechodzenia.

Twórcy serwisów WWW mogą za pomocą języka HTML 4 tworzyć własne kontrolki aplikacji lub *widgety*, jak są one nazywane w technologii ARIA. Używa się w tym celu elementów, które nie przejmują ogniska wprowadzania i które z racji tego nie mogą być wykorzystywane jedynie za pomocą klawiatury. Stosując technologię ARIA, można sprawić, aby wszystkie widoczne elementy miały przypisany **`tabindex`**, dzięki czemu mogą one przejmować ognisko wprowadzania i dzięki czemu można ich używać za pomocą klawiatury.

Ta możliwość technologii ARIA jest w tej chwili obsługiwana przez przeglądarki IE 8, Opera 9.5, Safari 4 oraz Firefox 3.5.

Role

W kodzie HTML do dostarczania semantycznej informacji o dokumentach wykorzystuje się nazwy elementów; mimo że sprytni twórcy serwisów WWW używają również atrybutów **class** i **id** do dołączania dodatkowych informacji na temat elementów, rozwiązanie to ma raczej charakter zwyczajowy i nie jest oficjalną częścią standardu. Wynika stąd, że język HTML jest nieco zubożały, jeśli chodzi o jego możliwości semantyczne. ARIA rozszerza możliwości standardu XHTML 1.1 związane z atrybutem **role**, pozwalając twórcom opisywać dodatkową funkcję wypełnianą przez element, niezwiązaną bezpośrednio z tym, jakiego rodzaju jest to element. Na przykład listy są elementami **li**, mogą też jednak odgrywać rolę nawigacyjną, choć i to nie musi być ich jedyna funkcja. Technologia ARIA dostarcza zbiór możliwych wartości atrybutu **role**, czyli zestaw nazw typów ról pełnionych przez element. Fachowo mówi się o tym, że zapewnia atrybutowi **role** *ontologię*.

Korzystając z atrybutu **role** do opisywania roli odgrywanej przez określony element, twórcy serwisów WWW mogą sprawić, aby przeglądarki „rozumiejące" role technologii ARIA lepiej przedstawiały użytkownikom strukturę dokumentu. Atrybut **role** może również zapewniać standardowy, przewidywany przez użytkownika sposób działania i wygląd określonych rodzajów kontrolek, na przykład pól wyboru, które są implementowane przez twórcę za pomocą innych niż typowe elementów HTML (typowym elementem byłby w tym przypadku **<input type="checkbox">**).

Role oferowane przez technologię ARIA należą do wielu różnych kategorii, wśród których wymienić można następujące:

- **Role znaków orientacyjnych**, które opisują regiony strony przeznaczone do celów nawigacyjnych. Należą do nich: **application**, **banner**, **main**, **navigation** oraz **search**.
- **Role struktury dokumentu**, które opisują funkcję odgrywaną przez element w strukturze dokumentu. Należą do nich: **navigation**, **section**, **note** oraz **heading** (jak przekonamy się w rozdziale 11., są one zwykle ściśle związane z nowymi elementami strukturalnymi standardu HTML 5).
- **Role struktury aplikacji**, które opisują funkcję odgrywaną przez element w strukturze aplikacji. Należą do nich: **alert**, **alertdialog**, **progressbar** oraz **status**.
- **Elementy interfejsu użytkownika**, takie jak **treegrid**, **toolbar** oraz **menuitem**.

- **Role elementów wejściowych użytkownika**, takie jak **`checkbox`**, **`slider`** oraz **`option`**.

Korzystanie z atrybutu **`role`** jest łatwe — polega po prostu na tym, że dodajemy jedną ze zdefiniowanych wartości roli jako wartość atrybutu **`role`** elementu, dokładnie tak, jak odbywało się to w przypadku wartości atrybutu **`class`**. Załóżmy na przykład, że chcemy użyć elementu **`input`** z **`type image`** jako przycisku. W celu zapewnienia odpowiedniego opisu funkcji elementu wejściowego powinniśmy po prostu dodać rolę **`button`**, tak jak zostało to pokazane poniżej:

```
<input type="image" alt="font-weight: bold" src="./images/
↪bold-unpressed.png" role="button">
```

Co przeglądarka zrobi z informacją, że dany element jest przyciskiem, pozostawia się już inwencji twórców programu. Informacja ta jest jednak obecna i możemy wyobrazić sobie, w jaki sposób przeglądarki i urządzenia asystujące mogą wykorzystać tę wiedzę nie tylko dla potrzeb osób niepełnosprawnych, lecz również wszystkich innych użytkowników korzystających z serwisu.

Stany i właściwości

Oprócz umożliwiania twórcom serwisów WWW opisywania ról odgrywanych przez elementy technologia ARIA pozwala też jawnie deklarować stany elementów, a więc na przykład to, czy dany element jest w tej chwili naciśnięty, jak również podawać innego rodzaju informacje związane z elementami. Stany dostarczają dynamicznych danych na temat elementów, czyli takich informacji jak to, czy pole wyboru jest zaznaczone, podczas gdy właściwości zapewniają informacje podstawowe, związane z samą naturą obiektów. Oznacza to w praktyce, że rozróżnienie pomiędzy właściwościami a stanami nie jest szczególnie ważne, ponieważ obydwa te typy informacji działają w bardzo podobny sposób. Należy tu jednak zaznaczyć różnicę występującą pomiędzy stanami i właściwościami a atrybutem **`role`**. Gdy korzystamy z atrybutu **`role`**, przypisujemy wartości do samego tego atrybutu, lecz istnieje wiele atrybutów stanów i właściwości, zaś każdy z nich posiada swoją własną nazwę — innymi słowy, nie istnieją atrybuty ARIA o nazwie **`state`** lub **`property`**. Nazwy atrybutów właściwości i stanów ARIA zawierają przedrostek **`aria-`**; przykładem może tu być stan **`aria-disabled`**, który może przyjmować wartości **`true`** i **`false`**, bądź też właściwość **`aria-`**
`↪flawto`, która wskazuje następny element w zalecanej kolejności odczytywania.

Wróćmy do naszego przykładu z elementem wejściowym. Gdyby rozpatrywanym elementem był przycisk, który może przyjmować dwa stany (wciśnięty i niewciśnięty), poinformować o stanie bieżącym moglibyśmy za pomocą stanu ARIA **aria-pressed**, tak jak zostało to przedstawione poniżej.

```
<input type="image" alt="font-weight: bold" src="./images/
↪bold-unpressed.png" role="button" aria-pressed="false">
```

Moglibyśmy też następnie zmienić stan **aria-pressed**, korzystając z kodu JavaScript, gdy element zostałby kliknięty lub odebrał zdarzenie związane z naciśnięciem klawisza *Enter*.

CSS i ARIA

Wszystkie nowoczesne przeglądarki internetowe, w tym również IE 8, umożliwiają nam formatowanie elementów w oparciu o ich właściwości i stany ARIA (a także wartości ich atrybutów **role**) za pomocą selektorów atrybutów. Na przykład aby zmienić kolor tła elementu w momencie, gdy jest on naciśnięty, użylibyśmy następującego kodu CSS:

```
[aria-pressed=true] {
   background-color: #cfb725;
}
```

Dodawanie ról, właściwości i stanów ARIA do naszych serwisów WWW lub aplikacji wymaga dodatkowego nakładu pracy, lecz nie jest to nakład o wiele większy niż poświęcony na zastosowanie wartości **class** i **id**, jeśli zdecydowalibyśmy się na to rozwiązanie do osiągnięcia tego samego efektu. Zamiast jednak zmagać się z koniecznością zapewniania własnego mechanizmu przechwytywania właściwości i stanów widgetu czy też innej części strony lub aplikacji, możemy tu po prostu ponownie wykorzystać ten doskonale przemyślany, zestandaryzowany sposób. Nie tylko pomoże nam to poprawić dostępność i ogólną użyteczność naszej części sieci WWW, lecz również umożliwi pisanie łatwiejszego w konserwacji i bardziej spójnego kodu.

Z uwagi na to, że twórcy serwisów WWW coraz chętniej używają bibliotek i platform, takich jak Dojo, JQuery, YUI oraz innych rozwiązań tego rodzaju, o których wspominaliśmy w rozdziale 5., jak również ze względu na fakt, że biblioteki te w coraz większym stopniu wspierają technologię ARIA, znaczna część pracy mającej na celu zapewnienie jej obsługi jest już wykonana za nas przez twórców tych bibliotek. Z efektów tych wysiłków możesz korzystać między innymi w następujących rozwiązaniach:

- JQuery — Podstawowy zespół programistów rozpoczął prace nad zapewnieniem wsparcia technologii ARIA i istnieje już wtyczka o nazwie jARIA, której zadaniem jest udostępnienie możliwości ustawiania i pobierania ról, właściwości i stanów za pośrednictwem kodu JavaScript i przy użyciu składni JQuery.
- Dojo — Zestaw narzędzi Dojo 1.0 zapewnia pełne wsparcie technologii ARIA w przypadku zbioru widgetów DojoX.
- YUI — Wiele widgetów YUI oferuje obsługę technologii ARIA.

Obsługa zapewniana przez przeglądarki i urządzenia asystujące

A zatem jaki poziom obsługi zapewniają współczesne przeglądarki internetowe możliwościom oferowanym przez technologię ARIA? Przekonaliśmy się już, że atrybut **`tabindex`** oraz możliwość przechwytywania ogniska wprowadzania przez każdy widoczny element obsługiwane są bardzo powszechnie, dotyczy to również programu Internet Explorer w wersji 5 i późniejszych. Oto jak mają się sprawy w przypadku innych możliwości tej technologii:

- IE 8 zapewnia obsługę ról, właściwości i stanów ARIA.
- Firefox 3.5 oferuje najpełniejszą obsługę technologii ARIA ze wszystkich współczesnych przeglądarek internetowych, w tym także obsługę ról, właściwości i stanów.
- Zgodnie z tym, co publikuje przedsiębiorstwo Opera Software, „Opera 9.5 obsługuje parsowanie ARIA w kodzie HTML… Wsparcie to ma charakter eksperymentalny, ponieważ standard ARIA się ustala”.
- Safari 4 oferuje ograniczoną obsługę technologii ARIA, umożliwiając korzystanie z wielu często używanych ról, nie zapewniając jednak wsparcia dla stanów czy właściwości. Z uwagi na fakt, że Safari radzi sobie z obsługą selektorów atrybutów CSS dla dowolnych atrybutów, niezależnie od tego, czy są one częścią bieżącego standardu HTML, czy też nie, możliwe jest formatowanie kodu HTML wyświetlanego za pomocą Safari przy użyciu wartości atrybutów właściwości i stanów ARIA. Dokładnie to samo odnosi się do Opery, Firefoksa 3.5, a nawet Internet Explorera.
- Być może najważniejsze jest to, że dwa najbardziej rozpowszechnione czytniki ekranowe, czyli Window-Eyes oraz JAWS, oferują solidne wsparcie dla technologii ARIA w przypadku swoich najnowszych wersji.

Przedstawione tu zaangażowanie twórców w implementację technologii ARIA wskazuje szerokie poparcie, jakie zdobyła ona sobie wśród osób związanych z WWW. Oprócz tego wiele bardzo popularnych serwisów i aplikacji WWW, w tym takie jak czytnik Google Reader firmy Google czy Gmail, intensywnie wykorzystuje tę technologię. W związku z tym twórcy mogą mieć pewność, że ARIA stanowi technologię, w którą zdecydowanie warto zainwestować swój czas i wysiłek niezbędny do jej opanowania.

ARIA i walidacja

Jak zatem możemy wykorzystywać technologię ARIA w naszym kodzie już dzisiaj? Wiemy, że proste dodawanie do naszego kodu atrybutów nienależących do specyfikacji języka HTML spowoduje, że opracowane w ten sposób dokumenty będą nieprawidłowe. Istnieje kilka metod korzystania z tych możliwości, które zapewniają jednocześnie poprawność dokumentów, ich zastosowanie może jednak wymagać nieco więcej niż tylko zwykłego wyboru odpowiedniego **DOCTYPE**.

Możliwe jest używanie własnych **DTD** (*definicji typu dokumentu* — ang. *document type definitions*), które zawierają atrybuty ARIA. Obecnie nie istnieją odpowiednie typy **DOCTYPE** dla kodu HTML lub XHTML, choć zwracano się już do organizacji W3C, aby je opracowała. Paciello Group, doskonale znana i bardzo poważana firma zajmująca się doradztwem związanym z zagadnieniami dostępności, zaproponowała eksperymentalną definicję typu dokumentu o nazwie HTML 4.01+ ARIA (więcej na ten temat znajdziesz pod adresem: *www.paciellogroup.com/blog/?p=107*), której można używać wraz z walidatorem kodu HTML oferowanym przez W3C.

Eksperymentalny walidator HTML 5 udostępniony pod adresem *www.validator.nu* również umożliwia sprawdzanie kodu ARIA i HTML 5. Zgłasza on błędy w przypadku natrafienia na typy **DOCTYPE** niezgodne ze standardem HTML 5, ale jest bardzo przydatnym narzędziem, dzięki któremu można zapewnić, że technologia ARIA jest używana w należyty sposób. Walidator ten ma pewne ograniczenia w kwestii aspektów technologii ARIA, które jest w stanie przetestować, a jednym z nich jest brak możliwości rozpoznawania ról „punktów orientacyjnych".

Rozwój technologii ARIA prowadzi jednak do znacznie ważniejszego pytania: czy walidacja zawsze powinna być istotnym celem? W końcu mimo potencjalnej olbrzymiej wartości technologii ARIA i mocnego wsparcia ze strony przeglądarek internetowych bez zapewnienia dobrej metody tworzenia prawidłowych dokumentów zawierających elementy ARIA technologia ta polegnie na przeszkodzie walidacji. Gdy mamy do czynienia z wykorzysta-

niem nowych technologii WWW, walidacja przypomina nieco prowadzenie samochodu przy patrzeniu wyłącznie we wsteczne lusterko. Henri Sivonen, jedna z bardziej wpływowych osób w procesie tworzenia języka HTML 5 (a także twórca walidatora kodu HTML 5), stwierdził:

> *Wykorzystanie nowych możliwości jest znacznie ważniejsze niż osiąganie zgodności ze starszymi celami walidacji. ARIA dodaje pewne znaczniki, dlatego nie spełnia wymogów walidacji starszych standardów, takich jak XHTML 1.0 (bez wstecznej modyfikacji tego, czym jest XHTML 1.0).* (cytat za stroną: *wiki.codetalks.org/wiki/index.php/Web_2.0_Accessibility_with_WAI-ARIA_FAQ*).

Dostępne są odpowiednie rozwiązania, jeśli walidacja jest bezwzględnie konieczna z uwagi na wymagania wewnętrzne lub regulacyjne. Jeśli jednak walidacja ma zasadniczo charakter mechanizmu zapewniania jakości, możliwe jest inteligentne wykorzystanie walidatorów kodu HTML 4 lub HTML 5 w celu przeprowadzenia kontroli jakości Twojego kodu zawierającego elementy ARIA nawet wówczas, gdy dokument nie jest ściśle zgodny z ich wymogami.

Typowe problemy związane z dostępnością (i ich rozwiązania)

Rozdział ten zakończymy opisem niektórych z najczęściej spotykanych problemów dotyczących dostępności oraz sposobów ich rozwiązywania wyłącznie za pomocą kodu HTML i CSS.

Łącza i tytuły

Łącza są intensywnie wykorzystywane przez czytniki ekranowe i urządzenia asystujące w celu umożliwienia użytkownikom szybkiego dostępu do zawartości strony. Mimo że stanowią bardzo niewielki procent tekstu znajdującego się na stronie, wielu użytkownikom oferują kluczową metodę dostępu do jej zawartości. Tekst łącza jest istotną wskazówką dotyczącą tego, dokąd to łącze prowadzi. W celu zapewnienia optymalnej dostępności tekst ten powinien „jasno identyfikować cel każdego łącza" (WCAG 1) i nie powinien zawierać w sobie jakichkolwiek założeń związanych z wykorzystywanym przez użytkownika urządzeniem wejściowym, co przejawiać się może w stosowaniu takich opisów łączy jak „kliknij tutaj". Gdy w większej liczbie łączy znajdujących się na stronie wykorzystuje się ten sam tekst, wszystkie one powinny wskazywać tę samą lokalizację.

Choć wiele osób uważa, że atrybut **title** związany z elementami łączy dostarcza dodatkowych informacji istotnych dla dostępności, w rzeczywistości wcale nie musi on być aż tak pomocny. Atrybut ten może w gruncie rzeczy wyrządzić pewne szkody, może bowiem przesłaniać inną treść w przypadku przeglądarek, które wyświetlają informację dostarczaną przez **title** w postaci podpowiedzi, nie jest dostępny, zanim użytkownik nie poruszy myszą, nie mogą też z niego korzystać użytkownicy czytników ekranowych, w których nie określono ustawień odpowiedzialnych za odczytywanie tych wartości. Przedstawiciele branży nie są zgodni w kwestii tego, czy należy używać atrybutu **title**, czy też raczej unikać jego stosowania, zdecydowanie jednak nie powinien on stanowić jedynego dostępnego dla użytkowników sposobu identyfikacji docelowego miejsca łącza.

Nagłówki

Czytniki ekranowe często wykorzystują nagłówki znajdujące się na stronie do tworzenia „spisów treści", które mają pomagać użytkownikom w jej przeglądaniu i przechodzeniu do odpowiednich obszarów. Jeśli stosujesz elementy nagłówków w roli *nagłówków* i używasz ich we właściwej kolejności, masz szansę sprawić, że ten sposób działania czytników ekranowych stanie się bardziej użyteczny. Nie pomijaj zatem poszczególnych poziomów nagłówków (a więc nie przeskakuj na przykład bezpośrednio z **h2** do **h4**). Style nagłówków widoczne na stronie można zawsze zmienić za pomocą mechanizmu CSS, dlatego nie ma potrzeby pomijania w kodzie któregokolwiek poziomu nagłówków.

Tekst alternatywny

Każda treść nietekstowa powinna mieć swój tekstowy zamiennik. W przypadku obrazów możesz go zapewnić, stosując atrybut **alt**. (Pedantyczna uwaga weterana zamierzchłych czasów: **alt** to nie znacznik). Wszelka treść dodawana za pomocą atrybutu **alt** powinna być zwięzła i — choć w przypadku wszystkich elementów **img** wymagane jest zastosowanie tego atrybutu w celu zapewnienia udanej walidacji strony — dla obrazów o charakterze czysto dekoracyjnym w roli wartości atrybutu **alt** powinien występować pusty ciąg znaków. Jeszcze lepiej jednak, gdy czysto dekoracyjne obrazy (a więc na przykład symbole pozycji listy czy ozdobne obrazy przeznaczone do roli teł) są dołączane za pomocą mechanizmu CSS, nie zaś w kodzie strony, ponieważ ten ostatni powinien być zarezerwowany wyłącznie dla treści użytecznej z semantycznego punktu widzenia.

Kontrast kolorów

Używanie dość nieprecyzyjnego terminu *niezdolności widzenia kolorów* prowadzi często do nieporozumienia polegającego na tym, że zwykłym ludziom może się wydawać, iż osoby z tą przypadłością widzą świat jedynie w odcieniach szarości. Ten rodzaj uszkodzenia wzroku (czyli „widzenie monochromatyczne") jest jednak w rzeczywistości bardzo rzadki; większość ludzi z upośledzeniem widzenia barw widzi w kolorze, ma jednak trudności z *rozróżnianiem* określonych kolorów, z rozróżnianiem których łatwo radzą sobie osoby niecierpiące na tego rodzaju dolegliwość.

Mimo że bardzo ważne jest unikanie korzystania z kolorów w celu komunikowania pewnych znaczeń (nie należy zatem na przykład stosować barwy czerwonej w roli ostrzeżenia, jeśli nie towarzyszy jej dodatkowy kształt, tekst lub informacja innego rodzaju), trzeba również rozważyć inne kwestie dostępności związane z kolorami. Powszechnie lekceważonym problemem jest brak wystarczającego kontrastu graficznego pomiędzy tekstem i jego tłem. Zestawienie tekstu i tła, które osobie widzącej prawidłowo wydaje się w „oczywisty sposób" mocno skontrastowane, może być zupełnie nieczytelne dla kogoś, kto cierpi na daltonizm lub inną wadę wzroku.

Za pomocą wielu narzędzi jesteśmy w stanie stwierdzić, czy para kolorów charakteryzuje się odpowiednio wysokim kontrastem, aby zapewnić czytelność, jednak ich używanie może być dość czasochłonne, ponieważ wymaga od Ciebie ręcznego porównywania każdej stosowanej w serwisie kombinacji barwy pierwszoplanowej z kolorem tła. Narzędzia takie jak AccessColor firmy AccessKeys (*www.accesskeys.org/tools/color-contrast.html*) umożliwiają analizowanie całych dokumentów poprzez przeglądanie skryptów DOM i wskazywanie potencjalnych problemów z kontrastem. Większość narzędzi do sprawdzania kontrastu radzi sobie jednak wyłącznie w sytuacjach, gdy tekst porównywany jest z kolorem tła (nie zaś z obrazem), a także nie testuje kontrastu w przypadku efektów dynamicznych, takich jak unoszenie, nie jest zatem w stanie zapewnić niezawodnych metod badania kontrastu barw.

Tabele

Tabele sprawiają szczególne problemy osobom ze słabym wzrokiem. Struktura tabeli, która wydaje się oczywista dla osób widzących prawidłowo, może być bardzo zagmatwana, gdy zostaje przedstawiona przez czytnik ekranowy. Język HTML oferuje szereg elementów i atrybutów, których zadaniem jest ułatwienie odczytywania i interpretowania zawartości tabel za pomocą czytników ekranowych.

Oprócz standardowych elementów **td** (danych tabeli) tabele mogą również zawierać elementy nagłówków tabeli (**th**), o czym przekonaliśmy się już w rozdziale 4. Aby w jak największym stopniu poprawić dostępność tabel, w przypadku komórek stanowiących nagłówki wierszy lub kolumn powinieneś korzystać z elementów **th**.

caption oraz summary

Zadaniem elementu tabeli **caption** jest dostarczanie krótkiego opisu tabeli, który jest wykorzystywany zarówno w sposób wizualny przez przeglądarkę, jak i przez czytniki ekranowe. Zaleca się, aby wszystkie tabele były opisane w ten sposób. Element **caption** powinien pojawić się bezpośrednio po otwierającym znaczniku tabeli. Tabele mogą mieć tylko jeden element tego typu.

```
<table>
<caption>Harmonogram pierwszego dnia konferencji</caption>
```

Oprócz niego element tabeli może zawierać atrybut **summary**, którego zadaniem jest dostarczanie przeglądu zawartości tabeli użytkownikom czytników ekranowych. Osoby widzące prawidłowo mogą szybko dowiedzieć się, jaki jest cel zamieszczenia tabeli i jaka jest jej zawartość; atrybut **summary** ma dostarczyć tej wiedzy również ludziom z problemami wzrokowymi. W przeciwieństwie do opisów definiowanych za pomocą elementów **caption** podsumowania określane przy użyciu atrybutów **summary** nie są renderowane przez przeglądarki internetowe — są one używane wyłącznie przez czytniki ekranowe.

```
<table summary="Pełny harmonogram wszystkich trzech ścieżek
↪pierwszego dnia konferencji Web Directions South 2009">
<caption>Harmonogram pierwszego dnia konferencji</caption>
```

abbr oraz scope

Do tej pory udało nam się już dostarczyć przeglądarce całkiem sporą ilość informacji, które sprawią, że nasza tabela będzie znacznie bardziej dostępna. Jednak zwłaszcza w przypadku skomplikowanych tabel język HTML oferuje dużą liczbę niedocenianych, lecz bardzo pomocnych możliwości, z których mogą korzystać czytniki ekranowe w celu ułatwienia użytkownikom właściwego zinterpretowania tabel danych.

Z uwagi na fakt, że czytniki ekranowe odczytują zwykle wszystkie komórki tabeli (w tym również komórki nagłówków) w sposób liniowy, każdy nagłówek jest odczytywany wielokrotnie, co może być dość czasochłonne i uciążliwe. Aby tego uniknąć, możesz zastosować atrybut **abbr** związany z ele-

mentem **th**. Pozwala nam to na zdefiniowanie skrótu tekstu, który ma być odczytywany zamiast pełnej treści elementu **th**.

```
<th abbr="projetkowanie">Ścieżka Projektowa</th>
<th abbr="menedżer">Ścieżka Menedżerska</th>
<th abbr="tworzenie">Ścieżka Twórcza</th>
```

Z samej struktury tabeli nie zawsze jasno wynika, czy element **th** jest nagłówkiem wiersza, czy też kolumny komórek. Atrybut **scope** umożliwia nam określenie, dla których komórek element **th** stanowi nagłówek. W przedstawionym powyżej przypadku nagłówki związane są z kolumnami, dlatego odpowiedni kod powinien mieć następującą postać:

```
<th abbr="projektowanie" scope="col">Ścieżka Projektowa</th>
<th abbr="menedżer" scope="col">Ścieżka Menedżerska</th>
<th abbr="tworzenie" scope="col">Ścieżka Twórcza</th>
```

W przypadku stosunkowo prostych tabel czytniki ekranowe mogą korzystać z atrybutu **scope** w celu odczytywania nagłówka (lub jego skrótu, jeśli zastosowaliśmy odpowiedni atrybut **abbr**) bezpośrednio przed zawartością komórki.

Na przykład na rysunku 6.1 widoczna jest tabela, w przypadku której czytnik ekranowy mógłby odczytać zaznaczoną komórkę jako: „projektowanie Osadzanie czcionek i typografia Marek Butelka". Jednak — jak doskonale pokazuje ten przykład — nadal jest to odległe od pełnego obrazu danych, które należałoby odczytać z tabeli. Informacja, którą czytnik powinien otrzymać, musiałaby raczej brzmieć „10:45 – 11:40 projektowanie Osadzanie czcionek i typografia Marek Butelka". W istocie z komórką należy tu połączyć więcej niż tylko jeden nagłówek. Możemy to zrobić, dodając atrybut **id** do każdego z nagłówków, a następnie korzystając z atrybutów **headers** związanych z elementami **td** w celu powiązania z nimi odpowiednich elementów **th**.

```
<table summary="Pełny harmonogram wszystkich trzech ścieżek
↪pierwszego dnia konferencji Web Directions South 2009">
<caption>Harmonogram pierwszego dnia konferencji</caption>

   <tr>
      <th>Godzina</th>
      <th abbr="projektowanie" scope="col">Ścieżka
      ↪Projektowa</th>
      <th abbr="menedżer" scope="col">Ścieżka Menedżerska</th>
      <th abbr="tworzenie" scope="col">Ścieżka Twórcza</th>
   </tr>

   <tr>
```

Rysunek 6.1. Osoba o dobrym wzroku może łatwo wywnioskować, że prezentacja dotycząca osadzania czcionek należy do ścieżki projektowej i rozpocznie się o godzinie 10:45. Dla użytkowników korzystających z czytników ekranowych z pewnością nie będzie to takie proste, chyba że zastosujemy odpowiednie znaczniki


```
   <th id="godzina1">7:00 – 9:00</th>
   <td colspan="3">Rejestracja</td>
</tr>

<tr>
   <th id="godzina2">9:00 – 9:10</th>
   <td colspan="3">Otwarcie</td>
</tr>

<tr>
   <th id="godzina3">9:10 – 10:15</th>
   <td colspan="3">Otwierająca prezentacja ogólna</td>
</tr>

<tr>
   <th id="godzina4">10:15 – 10:45</th>
   <td colspan="3">Poranna herbata</td>
</tr>

<tr>
   <th id="godzina5">10:45 – 11:40</th>
   <td headers="projektowanie godzina5"><a
   ↪href="http://south09...">Osadzanie czcionek...</a>
   <a href="http://south09...">Marek Butelka</a></td>
   <td headers="menedżer godzina5"><a
   ↪href="http://south09...">Dalej...></a></td>
```

```
    <td headers="tworzenie godzina5"><a
    ↪href="http://south09...">...</a></td>
</tr>
```

Choć zapewnienie tych dodatkowych informacji wymaga pewnego nakładu pracy związanego z kodowaniem, ich wpływ na dostępność strony WWW dla osób korzystających z czytników ekranowych może mieć bardzo istotne znaczenie. Podobnie jak większość sposobów tworzenia technika ta widziana i stosowana po raz pierwszy może Ci się wydawać zbędnym ciężarem, ale z czasem wykorzystywanie odpowiednich atrybutów i elementów stanie się Twoją drugą naturą.

Formularze

Bezpośrednio i w znacznym stopniu możesz poprawić dostępność swoich stron WWW przez zastosowanie odpowiednich znaczników w stosunku do zamieszczanych na nich formularzy. Wykorzystanie kilku prostych metod pozwoli Ci znacznie zwiększyć użyteczność formularzy dla osób z różnego rodzaju ograniczeniami.

Niezwykle istotnych informacji dla użytkowników czytników ekranowych dostarczają etykiety pól formularzy. Mimo że wskazówki graficzne znacznie ułatwiają określenie funkcji poszczególnych pól tym z nas, którzy dysponują dobrym wzrokiem, użytkownicy czytników ekranowych nie będą w stanie w prosty sposób stwierdzić, do czego służą pola formularza, jeśli nie zostaną one jawnie i prawidłowo połączone z odpowiednimi etykietami.

Niektóre elementy formularzy — na przykład elementy **button** — mają swoje własne etykiety. W ich przypadku wartość atrybutu **value** stanowi etykietę danego elementu. Jeśli nie istnieje żadna jawna etykieta, jak ma to miejsce w przypadku większości elementów formularzy, należy zastosować element **label** i za pomocą jego atrybutu **for** powiązać go bezpośrednio z elementem, dla którego ma on być etykietą. Poniżej zaprezentowany został sposób, dzięki któremu można zapewnić maksymalną dostępność w przypadku pola tekstowego posiadającego etykietę:

```
<label for="nazwisko" id="etykieta-nazwisko">Nazwisko</label>
↪<input type="text" id="nazwisko" aria-labelledby=
↪"etykieta-nazwisko">
```

Zwróć uwagę na fakt, że zastosowaliśmy tu również atrybut ARIA **aria-↪labelledby** w obrębie samego elementu formularza — a więc w elemencie **input** przedstawionym w poprzednim przykładzie — w celu poinformowania przeglądarki o **id** elementu, który jest opisywany przez bieżący

element. W przykładzie naszym wskazaliśmy zatem, że element (a dokładnie element **label**) z **id** o wartości **etykieta-nazwisko** stanowi etykietę dla elementu **input**, którego **id** to **nazwisko**.

Spostrzegawczy czytelnicy mogą też zauważyć, że atrybut **aria-labelledby** zachowuje się tak jak dostępny w standardzie HTML 4 atrybut **for**, działa jednak niejako w przeciwnym kierunku: w przypadku **for** to element etykiety identyfikuje element opisywany, wykorzystując do tego jego **id**, zaś w przypadku **aria-labelledby** to sam element opisywany wskazuje opisujący go element za pomocą *jego* **id**. Zwróć także uwagę na to, że nazwa **labelledby**, choć nie jest zgodna ze standardową, poprawną pisownią amerykańską, jest w tym przypadku jak najbardziej prawidłowa.

Gdy nie można zastosować elementu **label**, WCAG 2 sugeruje, że w roli etykiety elementu wystarczy użyć należącego do niego atrybutu **title**. Do skorzystania z **title** możemy być na przykład zmuszeni w przypadku pola wyszukiwania znajdującego się na pasku narzędzi strony, gdzie etykieta musiałaby zająć więcej miejsca w poziomie, niż jest to możliwe.

Grupowanie elementów pól

Zrozumienie bardziej skomplikowanych formularzy można ułatwić użytkownikom poprzez zgrupowanie związanych ze sobą pól i wyposażenie każdej z tych grup w odpowiednie nagłówki. Język HTML właśnie w tym celu oferuje elementy **fieldset** oraz **legend**. *Zbiór pól* jest po prostu elementem **fieldset**, który zawiera wszystkie związane ze sobą elementy formularza. Pierwszym elementem wchodzącym w skład **fieldset** powinien być element **legend**, który spełnia tu podobną funkcję jak opis w przypadku tabeli.

```
<fieldset>
   <legend>Dane do rozliczenia</legend>
   <p>Najpierw jednak musimy wiedzieć, <strong>kto zapłacić za
   ↪bilety?</strong></p>
   <ol>
      <li>
         <label for="billing_name"><em
         ↪class="required">*Nazwisko</em></label>...
</fieldset>
```

Sposoby, które tu opisaliśmy, pozwalają poradzić sobie z wieloma typowymi problemami dotyczącymi dostępności, z jakimi spotykają się użytkownicy internetu. Implementowanie ich nie jest zwykle zbyt uciążliwe, a znacznie poprawia wygodę korzystania z naszych serwisów WWW w przypadku osób,

które w największym stopniu są uzależnione od sieci. Sposoby te w dużej mierze pomagają również w spełnieniu ewentualnych wymagań stawianych przez regulacje prawne i instytucjonalne organizacji, dla której pracujesz.

Podsumowanie

W rozdziale tym ogólnie przyjrzeliśmy się kwestii dostępności, w tym również związanym z nią uwarunkowaniom prawnym i etycznym, istniejącym standardom W3C oraz występującym tu specyficznym problemom i ich rozwiązaniom. Co oprócz tego powinieneś wynieść z jego lektury? Najważniejszą ideą jest tutaj to, że prawdziwej dostępności nie da się osiągnąć jedynie poprzez wdrażanie listy odpowiednich poprawek na samym końcu procesu tworzenia serwisu WWW, lecz raczej w wyniku zastosowania holistycznego podejścia do projektowania i tworzenia treści WWW. Z tak pojętej dostępności korzystają nie tylko osoby niepełnosprawne, lecz również wszyscy użytkownicy efektów naszych działań. (Te same sposoby, które pozwalają poprawić dostępność treści, zwykle zwiększają także jej ogólną użyteczność).

Rola dostępności w tworzeniu serwisów WWW znacznie wzrosła w ostatnich dziesięciu latach, a odbyło się to w dużej mierze za sprawą pojedynczych misjonarzy sprawy działających w naszej branży. Trudno dyskutować z faktem, że uwarunkowania prawne i oczekiwania społeczne jak jeden mąż zmierzają w kierunku zapewniania większej dostępności dla coraz to większej liczby ludzi. Zamiast więc martwić się „kijem" w rękach ustawodawców i chętnych do wytyczania spraw prawników, pomyśl lepiej o „marchewce", czyli szansie dotarcia do wielu osób niepełnosprawnych, które odwiedzają Twój serwis, przyjazności dla mechanizmów wyszukiwarek internetowych cechującej kod zapewniający należytą dostępność, a także satysfakcji, która stanie się Twoim udziałem, gdy zrobisz coś, aby poprawić komfort korzystania z internetu tak wielu ludzi.