PRAWDA I TYLKO PRAWDA O HTML5!

Apress

HTML5

Wszystko, co powinniście wiedzieć o programowaniu

Przewodnik profesjonalisty

Luke Stevens, RJ Owen

Helion

Tytuł oryginału: The Truth About HTML5

Tłumaczenie: Piotr Rajca

ISBN: 978-83-246-9422-8

Original edition copyright © 2014 by Luke Stevens and RJ Owen.
All rights reserved.

Polish edition copyright © 2014 by HELION SA.
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/html5w
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

ROZDZIAŁ 4

Wszystko, co powinniście wiedzieć o strukturalnych elementach HTML5

Co udało nam się zrobić do tej pory?

- Sformułowaliśmy ogólny (i dosyć niejasny) pomysł, że strukturalne elementy HTML5 starają się coś poprawiać — zwłaszcza tworzenie planów dokumentów, które obecnie tworzy się niejawnie przy użyciu znaczników nagłówków.
- Określiliśmy, do czego służą plany dokumentów — wspomagają czytniki ekranowe, których działanie w znacznej mierze bazuje na wykorzystaniu nagłówków.
- Pobieżnie wspomnieliśmy także o lepszym sposobie wspomagania niewidomych użytkowników przeglądających nasze strony, czyli o użyciu punktów orientacyjnych ARIA.

Teraz nadszedł czas, żeby przyjrzeć się nieco dokładniej temu, co specyfikacja HTML5 (i to zarówno w wersji W3C, jak i WHATWG) ma do powiedzenia na temat nowych elementów strukturalnych. A zaczniemy od elementu…

<header>

O elemencie `<header>` trzeba wiedzieć dwie rzeczy:

- Element ten właściwie nic nie robi.
- Jego planowany sposób wykorzystania jest zapewne nieco odmienny od tego, co moglibyśmy sądzić.

Element `<header>` jest doskonałym przykładem, pokazującym, że powszechnie znany i używany termin ma w języku HTML5 zupełnie nowe znaczenie, a jednocześnie ma „utrwalać stosowane wcześniej rozwiązania”. Prawdopodobnie wielu z nas wciąż używa elementu `<div id="header">`, a zatem zastąpienie go elementem `<header>` powinno przynajmniej uprościć kod, prawda? Cóż, to właśnie jeden z tych momentów, gdy twórcy HTML5 pomyśleli: „Każdy tego używa, więc dlaczego by nie zmienić jego znaczenia?”. Oto, co specyfikacja ma do powiedzenia na temat tego elementu:

> *Element `header` reprezentuje grupę wprowadzających ułatwień nawigacyjnych.*
>
> *Uwaga: Zamierzonym sposobem użycia tego elementu jest umieszczanie wewnątrz niego nagłówków sekcji (takich jak elementy `h1` – `h6` czy też element `hgroup`), jednak nie jest to konieczne. Można go także używać do prezentacji spisu treści sekcji, formularza do wyszukiwania bądź też stosownych logo.*

Pouczająca jest uwaga: element `<header>` jest przeznaczony do tego, by umieszczać w nim *nagłówki sekcji*. Pamiętajmy, że w języku HTML5 sekcje są tworzone przy użyciu jednego z czterech elementów (`article`, `section`, `nav` oraz `aside`) i generują pozycję w planie dokumentu. Element `<header>` ma być umieszczany

Kup książkę Poleć książkę

wewnątrz elementów sekcji. Sam z siebie nie tworzy on odrębnej sekcji (choć wizualnie jest często przedstawiany w taki sposób, jakby tworzył) i nie jest dodawany do planu dokumentu.

A zatem o `<header>` należy myśleć jako o elemencie, który ma zawierać nagłówek sekcji. Może to zatem być cokolwiek, zaczynając do nagłówka głównej sekcji dokumentu, umieszczonego bezpośrednio w elemencie `<body>` (w takim przypadku element ten będzie zapewne zawierał logo oraz wszelkie inne elementy, które zazwyczaj traktujemy jako „nagłówek"), a kończąc na nagłówku komentarza.

Tak naprawdę to do niczego nie służy

Element `<header>` ma tylko jedno zastosowanie, mianowicie stwierdza: „To jest nagłówek tej sekcji". Problem polega na tym, że choć nasz kod może o tym informować, to żadna z przeglądarek ani żaden z agentów innych typów nie wydają się być tym zainteresowane.

Co więcej, zgodnie ze stwierdzeniem Hicksona: *nigdy się tym już nie zainteresują*. Hickson otwarcie przyznał, że wszystkie te nowe elementy mają głównie ułatwiać stosowanie stylów i nie powinny mieć większego wpływu na rozkład dokumentu (warto zapoznać się z jego komentarzem zamieszczonym w dalszej części rozdziału, pt. „Wniosek — świętej pamięci strukturalne znaczniki HTML5"). A zatem, element ten do niczego nie służy i najprawdopodobniej także w przyszłości nie będzie. Jest on semantycznym odpowiednikiem drzewa przewracającego się w lesie, gdy w pobliżu nie ma nikogo, kto by je usłyszał.

Zważywszy, że element `<header>` ani nie modyfikuje planu dokumentu, ani nie jest do niego dodawany, *faktycznymi* nagłówkami pojawiającymi się w tym planie (takimi jak `"Mój wspaniały blog"`) wciąż pozostają elementy `<h1>` – `<h6>`. A `<header>` istnieje tylko po to, by te elementy w nim umieścić, wraz ze wszystkimi innymi informacjami odpowiednimi dla nagłówka, takimi jak data. Przykładowy kod przedstawiający zastosowanie elementu `<header>` mógłby więc mieć następującą postać:

```
<header>
<h1>Mój kolejny wpis na blogu</h1>
<p>Opublikowany ... </p>
</header>
```

Jak czytniki ekranowe mogą używać elementu <header>, skoro może się on pojawiać wszędzie?

Można by sądzić, że czytniki ekranowe powinny pomijać elementy `<header>` i przechodzić bezpośrednio do ich zawartości. Jednak nie możemy mieć pewności (ani my, ani czytniki ekranowe), że pierwszy element `<header>` odnaleziony w dokumencie będzie reprezentował główny nagłówek strony. Jeśli kod dokumentu został zapisany w niestandardowej kolejności (na przykład najpierw jest zapisana treść, a potem nagłówek, stopka i pasek boczny), to dokument może zawierać kilka elementów `<header>`, z których żaden nie będzie pełnił funkcji typowego nagłówka. W ten sposób, pod względem obsługi nagłówków na stronie, wróciliśmy do punktu wyjścia.

Alternatywa ARIA — banner

Na szczęście dla niewidomych użytkowników, istnieje alternatywne rozwiązanie. Punkt orientacyjny ARIA `banner` wyznacza nagłówek w takim znaczeniu, w jakim potocznie go rozumiemy. Oto, w jaki sposób specyfikacja ARIA go definiuje (*http://www.w3.org/TR/wai-aria/roles#banner*):

> *Region, którego treść odnosi się głównie do witryny, a nie do strony.*
>
> *Do treści odnoszącej się do witryny można zaliczyć takie elementy jak logo lub znak reprezentujący sponsora witryny oraz narzędzie służące do przeszukiwania zawartości witryny. Nagłówek[1] jest zazwyczaj umieszczany na samej górze strony i przeważnie zajmuje całą jej szerokość.*

[1] ang. *banner — przyp. tłum.*

Kup książkę Poleć książkę

Taki nagłówek powinien się pojawiać tylko raz w całym dokumencie, tak by czytniki ekranowe mogły przeskoczyć bezpośrednio do niego i by mogły mieć dużą pewność odnośnie tego, co on reprezentuje — i dokładnie o to nam chodzi.

Rekomendacja

Przeznaczenie elementu `<header>` jest zbyt szerokie (i zbyt bezcelowe), by zapewnić jego przydatność. Zamiast niego warto stosować rozwiązanie ARIA, atrybut `role="banner"`, umieszczany w odpowiednim elemencie (a w razie konieczności w Internet Explorerze 6 także nadmiarową klasę `"banner"`).

<nav>

Oto, co na jego temat można znaleźć w specyfikacji:

> *Element nav reprezentuje sekcję strony, zawierającą odnośniki do innych stron lub miejsc w ramach danej strony: sekcję z elementami nawigacyjnymi.*
>
> *Nie wszystkie grupy odnośników znajdujące się na stronie muszą być umieszczane w elementach nav — elementy nav powinny zawierać wyłącznie główne bloki nawigacyjne. W szczególności dotyczy to stopek, które powszechnie zawierają odnośniki do innych stron, takich jak regulamin, strona główna oraz informacje o prawach autorskich. W takich przypadkach wystarczy zastosować jedynie element `footer` — element nav nie jest konieczny.*

Element `<nav>` nie wyznacza nowej sekcji dokumentu i ma następujące zalety:

- Jego przeznaczenie jest stosunkowo jasne.
- Z pozoru jest całkiem użyteczny.

Chodzi o to, że jeśli umieścimy elementy nawigacyjne w elemencie `<nav>`, to osoby niewidome będą go mogły pominąć i przejść bezpośrednio do treści strony, a później, gdy zechcą przejść na inną stronę, przeskoczyć bezpośrednio do elementów nawigacyjnych.

Z punktu widzenia dostępności stron to przecież świetne rozwiązanie, prawda?

Dobra intencja, lecz tragedia pod względem dostępności

Choć intencje są słuszne, to jednak próba ułatwienia życia jednej mniejszości użytkowników potencjalnie może doprowadzić do jej utrudnienia innej mniejszości — osobom korzystającym z przeglądarek IE 6, 7 oraz 8, w których została wyłączona obsługa języka JavaScript.

Ze względu na sposób, w jaki te przeglądarki obsługują „nieznane” elementy HTML, nie są w nich stosowane żadne style CSS. Problem ten może dotknąć jednego użytkownika na 100, co jest większym odsetkiem niż liczba osób korzystających z czytników ekranowych. A to sprawia, że cała idea korzystania z elementu `<nav>` staje się nieco problematyczna. (Ten problem dotyczy także wszystkich pozostałych elementów HTML5 opisanych w dalszej części rozdziału). Wiele nowoczesnych platform stara się rozwiązać ten problem, używając języka JavaScript, by skłonić przeglądarki do potraktowania nowych znaczników jako czegoś, co są w stanie obsługiwać, jednak nie da się tego osiągnąć w starszych przeglądarkach, w których wyłączono obsługę języka JavaScript. Ponieważ te nowe znaczniki w ogóle nie będą działać w starszych przeglądarkach, zatem nie są w stanie spełnić jedynego celu, do którego, według Hicksona, mają służyć — poprawiania dostępności stron.

Kup książkę Poleć książkę

Alternatywa ARIA — navigation

Na szczęście, zamiast z elementu `<nav>` możemy skorzystać z punktu orientacyjnego ARIA `navigation` — dodanie do wybranego elementu `<div>` (lub `<ul>`) atrybutu `role="navigation"` pozwala poprawić dostępność strony bez jednoczesnego pogarszania jej dla innej grupy użytkowników. Specyfikacja ARIA definiuje punkt orientacyjny `navigation` (*http://www.w3.org/TR/wai-aria/roles#navigation*) w następujący sposób:

> *Kolekcja elementów nawigacyjnych (zazwyczaj odnośników) służących od poruszania się w obrębie danego dokumentu oraz dokumentów z nim powiązanych.*

Rekomendacja

Najlepiej jest używać atrybutu `role="navigation"`. Element `<nav>` należy uważać za szkodliwy, aż do momentu gdy liczba użytkowników korzystających z przeglądarki IE8 stanie się bardzo niewielka. (Odnosi się to do użytkowników systemu Windows XP, gdyż IE8 jest ostatnią przeglądarką instalowaną w tym systemie. Może to zająć trochę czasu).

<section> oraz <article>

W przypadku tych elementów sytuacja wygląda podobnie (każdy ma problemy z ich zrozumieniem), jednak ich sugerowane zastosowania różnią się od siebie. Najpierw przedstawimy każdy z tych elementów osobno, a następnie przeanalizujemy ich podobieństwa. Proszę, by podczas prób zrozumienia tych dwóch elementów powstrzymać się od rzucania przedmiotami oraz małymi zwierzątkami.

<section>

Oto fragment specyfikacji:

> *Element `section` reprezentuje ogólną sekcję dokumentu lub aplikacji. W tym kontekście sekcją jest fragment tematycznie powiązanych ze sobą treści, zazwyczaj zawierający także nagłówek.*
>
> *Przykładami sekcji mogą być rozdziały, karty w oknach dialogowych z kartami bądź też numerowane sekcje wypracowania. Strona główna witryny może być podzielona na takie sekcje jak: wprowadzenie, nowe elementy oraz informacje kontaktowe.*
>
> *Uwaga: Zachęca się autorów do stosowania elementów `article` zamiast `section` wszędzie tam, gdzie może wchodzić w grę łączenie treści elementów.*
>
> *Uwaga: Element `section` nie jest elementem pojemnika o ogólnym charakterze. Wszędzie tam, gdzie potrzebny jest element do zastosowania stylów lub ułatwiający działanie skryptów, autorzy powinni używać elementu `div`. Ogólna zasada określa, że element `section` powinno się stosować tylko w tych przypadkach, gdy treść elementu będzie jawnie wyświetlona w planie dokumentu.*

Spróbujmy coś z tego zrozumieć. Element `<section>` ma reprezentować ogólną sekcję dokumentu. A zatem, gdyby ten rozdział książki był stroną WWW, to moglibyśmy go podzielić na fragmenty, stosując do tego celu właśnie znaczniki `<section>`. Element ten może także reprezentować różne obszary strony głównej, od najnowszych doniesień aż do informacji kontaktowych. Jednak nie należy go używać jako ogólnego pojemnika, służącego jedynie do stosowania stylów — to wymaga użycia elementu `<div>`.

Oprócz tego, elementu `<section>` nie należy używać do tworzenia głównego obszaru treści strony (podobnie zresztą jak elementu `<article>`), jednak tym zagadnieniem zajmiemy się niebawem, podczas rozważań na temat brakującego elementu `<content>`.

Kup książkę Poleć książkę

Sekcje == punkty planu

Także w tym przypadku zrozumienie elementu `<section>` wymaga zrozumienia sposobów tworzenia planów dokumentów oraz podziału dokumentów na sekcje. Specyfikacja wspomina o tym (proszę spojrzeć na ostatnie zdanie drugiej uwagi), choć informacje te są raczej skąpe, zważywszy, że element `<section>` jest podstawą tworzenia planów dokumentu. Ogólna zasada głosi, przynajmniej jeśli chodzi o tworzenie planów dokumentów, że jeśli w danym miejscu nie można użyć elementu `<article>`, `<nav>` lub `<aside>`, to zapewne trzeba tam będzie użyć elementu `<section>`.

To właśnie także z tego powodu elementu `<section>` nie należy używać jako ogólnego pojemnika ułatwiającego stosowanie stylów. Jeśli zastosujemy go byle gdzie, tylko po to, by móc określać style, *bez* zwracania uwagi na tworzony w ten sposób plan dokumentu, to w efekcie będzie on całkowicie nielogiczny, co z kolei przekreśla całą ideę użycia znacznika `<section>`. To powszechny błąd, pokazujący, jak słabo przeznaczenie tych nowych znaczników HTML5 jest tłumaczone w dokumentacji, propagowane przez ekspertów i rozumiane przez społeczność (nie żebym ją o to winił).

Matrioszki

Nie zapominajmy, że sekcje można zagnieżdżać (niezależnie od tego, czy są one tworzone przy użyciu elementu `<section>`, `<article>`, `<nav>` czy też `<aside>`). Jak mieliśmy okazję przekonać się w rozdziale 3., w świecie czystego języka HTML5 to właśnie ten poziom zagnieżdżenia ma określać prawdziwy poziom nagłówków `<h1>` – `<h6>`, a nie zastosowany element nagłówka. W HTML5 przeglądarka (teoretycznie) powinna je traktować jako ogólne elementy nagłówka, o ile tylko zostaną umieszczone wewnątrz sekcji.

A zatem, powinniśmy móc wszędzie używać elementu `<h1>`, a przeglądarka powinna być w stanie określić, czy zostały one umieszczone w kodzie jako element `<h1>` czy jako element `<h101>`. Niemniej jednak, ze względu na czytniki ekranowe (i to zarówno te dostępne obecnie, jak i te, które będą używane w całkiem odległej przyszłości), konieczne będzie odpowiednie stosowanie nagłówków `<h1>` – `<h6>`, niezależnie od używanej wersji języka HTML.

Rekomendacja

Jeśli zależy nam na tworzeniu planów dokumentów w sposób zalecany w języku HTML5, to będziemy do tego celu używali głównie elementów `<section>`. Ponad 20 lat trwało, zanim element ten trafił do specyfikacji języka HTML (proszę sobie przypomnieć komentarz Tima Bernersa-Lee zamieszczony w poprzednim rozdziale), i zapewne drugie tyle potrwa, nim twórcy stron nauczą się go prawidłowo używać.

Element ten nie ma żadnego odpowiednika w specyfikacji ARIA.

<article>

Można by sądzić, że „artykuł”[2] na stronie WWW będzie odpowiadał „artykułowi w prasie”. Jednak wszyscy, którzy sądzili, że nowe elementy HTML5 mogą mieć intuicyjne zastosowanie, powinni się wstydzić. W tym przypadku jest to raczej „artykuł odzieżowy”. O tak… to kolejny „semantyczny” termin o bardzo nieintuicyjnym znaczeniu.

A oto fragment specyfikacji dotyczący tego elementu:

> *Element `article` reprezentuje zamknięty fragment kompozycji dokumentu, stronę, aplikację lub witrynę, które w zasadzie mogą być niezależnie rozpowszechniane lub wielokrotnie stosowane, na przykład w ramach łączenia treści. Może to być wpis na forum, artykuł w czasopiśmie lub gazecie, wpis na blogu, komentarz wpisany przez użytkownika, interaktywny widżet lub jakikolwiek inny niezależny element treści.*
>
> *W przypadku zagnieżdżania wewnętrzne elementy `article` reprezentują artykuły, które w zasadzie są powiązane z treścią zewnętrznego elementu `article`. Na przykład wpis na blogu, do którego można*

[2] ang. *article — przyp. tłum.*

Kup książkę Poleć książkę

dodawać komentarze, może je przedstawiać przy użyciu elementów `article` zagnieżdżonych wewnątrz elementu `article` reprezentującego sam wpis.

A oto, co na początku 2012 roku Hickson napisał na temat tego elementu na liście dyskusyjnej WHATWG (*http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2012-January/034506.html*):

Element <article> obejmuje szeroki zakres znaczeń:
- wpisy na forum,
- artykuły w gazetach,
- artykuły w czasopismach,
- książki,
- wpisy na blogach,
- komentarze do wpisów na forum,
- komentarze do artykułów w gazetach,
- komentarze do artykułów w czasopismach,
- komentarze do wpisów na blogach,
- osadzane na stronach interaktywne widżety,
- wpisy z fotografiami na witrynach społecznościowych,
- komentarze do fotografii na witrynach społecznościowych,
- specyfikacje,
- wiadomości poczty elektronicznej,
- odpowiedzi na wiadomości poczty elektronicznej.

W języku HTML 4 akapit to akapit i tylko akapit. W HTML5 „artykuł" jest wpisem na forum, który jest komentarzem do wpisu na blogu, który jest widżetem, który jest właściwym artykułem. Jeśli element ma tak szerokie znaczenie, w jaki sposób może być bardziej „semantyczny"? Chciałbym, żeby ustalono, że noże, łyżki, widelce i telewizory będą nazywane „widelcami".

To nie jest utrwalanie stosowanych wcześniej rozwiązań.

Także ten element najlepiej jest rozumieć w kontekście planów dokumentów. Służy on do tworzenia sekcji dokumentu w tych wszystkich przypadkach, kiedy nie chcemy używać elementu <section>, czyli zazwyczaj wtedy, gdy chcemy w nim umieścić jakiś fragment treści (bądź też „interaktywny widżet").

Specyfikacja wspomina, że element <article> może być używany w przypadkach łączenia treści, gdyż reprezentuje pewną niezależną całość, jednak kiedy i jak należałoby z tej możliwości korzystać, pozostaje niejasne (chyba lepiej będzie używać RSS!). To rozwiązanie, które aż prosi się o problemy.

Specyfikacje powinny precyzować

Podstawowy problem związany z elementem <article> polega na tym, że pozostawia on szerokie pole do interpretacji („Co oznacza »w zasadzie«? Nadający się do wielokrotnego użycia?").

Specyfikacje zawodzą, jeśli pozostawiają nam możliwość interpretacji. Ich celem jest precyzyjne określenie, co należy robić. Jednak w tym przypadku specyfikacja jest otwarta na interpretacje, nie daje żadnych jasnych korzyści i powtarza istniejące już możliwości funkcjonalne (element <article> to w sumie element <section> ze zmienioną nazwą).

Zagnieżdżanie elementów <article> w celu tworzenia artykułów i komentarzy

Elementy <article> można także zagnieżdżać wewnątrz innych elementów <article>, o ile tylko ich treść jest ze sobą powiązana. Specyfikacja sugeruje, że komentarze do wpisów na blogu mogą być zapisywane jako elementy <article>, a następnie umieszczone wewnątrz jednego zbiorczego elementu <article> reprezentującego

Kup książkę Poleć książkę

cały wpis. To problem przypominający nieco twierdzenie, że „wszystkie łyżki i widelce są widelcami". Jeśli dochodzi do zastosowania elementów `<article class="post">` oraz `<article class="comment">`, to `article` staje się jedynie nieco bardziej rozbudowaną formą elementu `div`.

Dlaczego nie można by po prostu dodać elementu `<comment>` i mieć w ten sposób możliwość odtworzenia standardowego wzorca artykułu z umieszczonymi poniżej komentarzami, który jest stosowany niemal na wszystkich istniejących w internecie blogach i witrynach zawierających jakieś publikacje? Czy właśnie to nie byłoby przykładem utrwalania stosowanych wcześniej rozwiązań? Przeciwnie do tego, co uważa Ian Hickson, który próbuje narzucić swój ekscentryczny pogląd na tę sprawę, twierdząc, że nie ma żadnej różnicy pomiędzy artykułem i komentarzem (*http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2012-January/034506.html*):

> *Sądzę, że opinia uznająca, że wypowiedzi autora witryny w jakiś sposób odróżniają się od wypowiedzi jej czytelników, jest anachronizmem. Na czym miałaby polegać taka różnica?*
>
> *Wprost przeciwnie, na WWW taka różnica _nie_ istnieje. Artykuł jest jedynie komentarzem podniesionym na nieco bardziej wyróżniającą się pozycję.*

Hickson najwyraźniej nie zauważył, jak ironiczne jest wymienienie co najmniej jednej różnicy przy jednoczesnym stwierdzeniu, że żadna różnica nie istnieje.

Oczywiście, stoi to także w jawnej sprzeczności z często wskazywanym przeznaczeniem tych elementów, którym ma być pomoc autorom w utrzymaniu ich dokumentów. Gdyby kiedyś mógł powstać wzorzec zastosowania kodu reprezentujący artykuł z komentarzami — to by było to! Jednak Hickson, działając jako oskarżony i sędzia, nie ustąpi, potwierdzając jedynie swoją dziwną, filozoficzną opinię, że wszystkie „komentarze" są sobie równe i kropka. Zostaniemy zatem z zagnieżdżanymi elementami `<article>` i można sądzić, że za jakiś czas cała zawartość WWW znajdzie się w tych elementach.

Wyszukiwarki nie potrzebują elementów <article>

Niektórzy mogą sądzić, że element `<article>` pomaga wyszukiwarkom, jednak one wcale go nie potrzebują, by wiedzieć, gdzie jest treść strony. Całe ich istnienie sprowadza się do tego, że są w stanie odnajdywać treści bez pomocy tego rodzaju. Nawet gdyby element `<article>` był powszechnie używany, to w jaki sposób na podstawie samego kodu przeglądarki miałyby się zorientować, czy dany element `<article>` reprezentuje wpis na blogu, wpis na forum, interaktywny widżet, komentarz czy też cokolwiek innego? Element ten ma zbyt ogólne znaczenie, by mógł być przydatny do celów SEO, i to nawet gdyby wyszukiwarki *zwracały* uwagę na to, jakich znaczników używamy w naszych dokumentach (a w przeważającej większości przypadków tego nie robią).

(Zagadnieniami związanym z językiem HTML5 i technikami SEO zajmiemy się bardziej szczegółowo w rozdziale 6.).

Element <article> nie ma także zawierać głównej treści strony

Element `<article>` nie powinien być także używany do reprezentacji głównego obszaru treści strony. Warto zauważyć, że gdy Hickson go wymyślał, *nie istniał* jeszcze żaden element mający takie przeznaczenie, a Hickson gwałtownie się sprzeciwiał jego powstaniu. Element `<main>` jeszcze nie istniał, a `<article>` bez wątpienia nie miał się nim stać.

Rekomendacja

Czy należy używać tego znacznika? Ja bym tego nie robił. Zamiast tego zaliczyłbym go do kategorii: winny-dopóki-nie-udowodni-swojej-niewinności. Jednak jeśli tylko pojawią się jakieś pragmatyczne korzyści wynikające z jego stosowania, to czemu nie! Jednak póki to nie nastąpi, to lepiej sobie odpuścić.

Podobnie jak `<section>`, także element `<article>` nie ma swojego odpowiednika w specyfikacji ARIA.

Kup książkę Poleć książkę

A zatem, jaka jest różnica pomiędzy elementami <article> i <section>?

Oto kilka rzeczy, które należy wiedzieć:

- Artykuły można zagnieżdżać w innych artykułach.
- Artykuły mogą być dzielone przy użyciu sekcji.
- Sekcja może być podzielona na artykuły, które z kolei mogą być podzielone na sekcje.
- Ludzie zupełnie nie potrafią spójnie używać znaczników.

Wiecie co? Za wyjątkiem garstki HTML-owych superbystrzaków, każdy, kto spróbuje używać tych znaczników (choć zdziwiłbym się, gdyby ktokolwiek to robił), utworzy tylko jeden wielki bałagan. Ale co tam, w końcu mogę się mylić.

Osobiście wolałbym raczej zarabiać na życie przy pomocy zardzewiałego nożyka do obierania ziemniaków, niż debatować nad zaletami znacznika `<section>` w porównaniu ze znacznikiem `<article>`. Sam fakt, że taka debata się toczy, dobitnie pokazuje, że specyfikacja zawodzi. Jeśli musimy zastanawiać się nad zastosowaniem znacznika, oznacza to, że przegraliśmy.

Jednak na temat różnic między tymi dwoma znacznikami napisano już tysiące słów na wszelkiego rodzaju blogach (jednym z przykładów jest wpis Bruce'a Lawsona: *http://www.brucelawson.co.uk/2010/html5-articles-and-sections-whats-the-difference/*) i forach dyskusyjnych.

Można by mieć nadzieję, że absurdalność sytuacji, która zmusza do tworzenia przydatnych, lecz pod każdym innym względem *śmiesznych* schematów blokowych (*http://html5doctor.com/downloads/h5d-sectioning-flowchart.png*) tylko po to, by określić, jaki element HTML5 należy zastosować, zmusi społeczność do zastanowienia. Najwidoczniej jednak tak się nie stało. A wszystko to z powodu decyzji jednego lub dwóch członków grupy WHATWG o dorzuceniu w 2004 roku tego dodatkowego elementu `<section>` do specyfikacji Web Application 1.0.

No dobrze, udało się nam okantować Band-Aid i przetrwać najbardziej bolesny fragment spotkania z nowymi elementami HTML5. Wciąż jednak pozostają pewne lepkie osady, które będą cuchnąć, gdy wyjdą na wierzch. Przyjrzymy się zatem dwóm ostatnim elementom sekcji.

<aside>

Pytanie: jak nazywamy wyróżniony z tekstu komentarz, notatki na marginesie lub pasek boczny?

Jeśli odpowiecie, że są to wyróżnione komentarze, notatki na marginesie lub pasek boczny, to przegraliście — trafiony, zatopiony. To przecież „aside".

To przecież było oczywiste, prawda? A oto, co na temat tego elementu ma do powiedzenia specyfikacja:

> *Element `aside` reprezentuje sekcję strony zawierającą treści, które można uznać za niezależne od treści otaczających ten element, ponieważ są z nimi marginalnie związane. W materiałach drukowanych takie treści są często reprezentowane w formie pasków bocznych.*
>
> *Element ten może być używany do tworzenia efektów typograficznych, takich jak wyróżnione komentarze lub paski boczne, do celów reklamowych, grupowania elementów nawigacyjnych, jak również do prezentowania treści, które są uważane za niezależne od podstawowej zawartości strony.*

Niełatwo jest zrozumieć, dlaczego wyróżnione komentarze, materiały reklamowe czy też elementy nawigacyjne (uzupełnione elementem `<section>` prezentującym zestaw odnośników do innych blogów oraz archiwum, bo taki przykład zastosowania jest przedstawiony w specyfikacji) należy nazywać w jeden i ten sam sposób. Jednak w dziwacznym świecie strukturalnej semantyki języka HTML5 tak właśnie jest, a my, na szczęście, możemy radośnie to zignorować.

Kup książkę Poleć książkę

Element <aside> tworzy sekcje w dziwnych miejscach

Trzeba pamiętać, że element `<aside>` tworzy sekcję w planie dokumentu, co jest tym bardziej dziwne, gdy uwzględnimy jego szeroki zakres zastosowań. (Niby dlaczego wyróżnione komentarze mają być umieszczone w odrębnych sekcjach?)

Gdyby element ten faktycznie służył do prezentowania pasków bocznych lub gdyby nazwano go `<sidebar>` (a początkowo właśnie tak się nazywał), to mógłby mieć sens, jednak zarówno jego przeznaczenie, jak i nazwa są inne, dlatego nie ma w nim także sensu.

Alternatywa ARIA — complementary

ARIA udostępnia alternatywę dla elementu `<aside>`, a jest nią punkt orientacyjny `complementary`. W specyfikacji został on opisany w następujący sposób (*http://www.w3.org/TR/wai-aria/roles#complementary*):

> *Sekcja dokumentu, pełniąca rolę wspomagającą i stworzona na podobnym poziomie DOM w celu uzupełniania głównej treści strony, a przy tym posiadająca sens i znaczenie, nawet gdyby została od tej głównej treści oddzielona.*
>
> *Istnieje wiele typów treści, które pozwalają na prawidłowe zastosowanie tej roli. Na przykład w przypadku portalu mogą one prezentować godziny, aktualną pogodę, powiązane artykuły lub kursy akcji, choć oczywiście możliwości zastosowania tej roli na tym się nie kończą. Rola `complementary` oznacza, że zawartość elementu jest istotna dla głównej treści strony. Gdyby jednak zawartość takiego elementu mogła zostać całkowicie odseparowana od głównej treści strony, to prawdopodobnie należałoby zastosować rolę o bardziej ogólnym charakterze.*

Rekomendacja

Sugeruję, by paski boczne w szablonach projektowanych stron tworzyć przy użyciu atrybutu `role="complementary"` dodawanego do odpowiednich elementów `<div>` (lub dowolnych innych).

<footer>

Czy pamiętacie, jak oczywisty wydawał się początkowo element `<header>`, a później okazało się, że wcale taki nie jest? Cóż, dokładnie tak samo jest w przypadku elementu `<footer>`. Można sądzić, że reprezentuje on główną stopkę strony, jednak w rzeczywistości można go używać do tworzenia stopki dowolnej sekcji. Oto informacje na jego temat podane w specyfikacji HTML5:

> *Element `footer` reprezentuje stopkę swojego najbliższego nadrzędnego elementu sekcji lub głównej sekcji całego dokumentu. Zazwyczaj zawiera on informacje dotyczące danej sekcji, takie jak jej autor, odnośniki do powiązanych dokumentów, informacje o prawach autorskich i tak dalej.*
>
> *Informacje o kontakcie z autorem lub redaktorem należy zapisywać w elemencie `address` i zapewne umieszczać w elemencie `footer`.*
>
> *Element `footer` wcale nie musi być wyświetlany na samym końcu sekcji, choć zazwyczaj właśnie tam się go umieszcza.*

Stopki, podobnie jak nagłówki, to części umieszczane wewnątrz sekcji — same z siebie nie tworzą odrębnych sekcji dokumentu. Także w tym przypadku panuje zamieszanie, gdyż przykłady prezentowane w książkach i wpisach na blogach poświęconych HTML prezentują element `<footer>` w taki sposób, jak gdyby tworzył odrębną *wizualną* sekcję o takim samym znaczeniu jak sekcje tworzone przez elementy `<article>` oraz `<aside>`. Jednak w rzeczywistości element ten w ogóle nie jest uwzględniany w planie dokumentu. Jego znaczenie polega jedynie na opisywaniu fragmentu nadrzędnego sekcji, niezależnie od tego, gdzie (i jak często) będzie się ona pojawiać w dokumencie.

Kup książkę Poleć książkę

Także stopki do niczego nie służą

Element `<footer>`, podobnie jak i `<header>`, w zasadzie nic nie robi. Okazuje się, że nie trzeba go nawet umieszczać na końcu elementu nadrzędnego. W specyfikacji jest zamieszczony przykład sekcji `<article>`, w którym metadane dotyczące komentarza (jego autor oraz data publikacji) są zapisane w elemencie `<footer>` i umieszczone — wyobraźcie to sobie — na *samej górze* komentarza. Najwyraźniej element `<footer>` nie jest jeszcze dostatecznie mylący.

I ponownie kłania się zagadnienie utrwalania wcześniej stosowanych rozwiązań. Ja na pewno nie słyszałem tysięcy projektantów i twórców stron głośno domagających się elementu `<footer>`, który można by dodawać do każdej sekcji na stronie. A Wy?

Obszerna stopka? Powodzenia!

A co, jeśli chcemy stworzyć nowomodną „obszerną stopkę", zawierającą grupę odnośników i innych informacji? Cóż, to całkiem sporo treści, więc powinna to być sekcja. Można by ją zatem utworzyć, używając jednego z elementów sekcji: `<nav>`, `<section>` lub nawet `<aside>`.

Jednak, żeby nie było tak nudno, taką sekcję można umieścić wewnątrz elementu `<footer>`. A zatem, element ten może oznaczać zarówno treść *należącą do* sekcji uwzględnianej w planie dokumentu, *jak i* zawierać taką sekcję, samemu jej przy tym nie tworząc.

Jasne jak deszczowa noc, prawda? Nie rozumiem, czemu tak jest, i nie sądzę, by rozumiała to większość projektantów i twórców stron WWW. Widziałem pokazy japońskich gier, które były bardziej sensowne niż te elementy.

Czy mogę prosić o stopkę?

Propozycje elementu `<footer>` pojawiały się już od dłuższego czasu. Kiedy w 2002 roku — czyli ponad dekadę temu — dyskutowano na jego temat, na publicznych listach dyskusyjnych W3C w kontekście języka XHTML 2.0 szybko zaczęto go krytykować (*http://lists.w3.org/Archives/Public/www-html/2002Aug/0257.html*), stwierdzając, że nie zapewni żadnych konkretnych korzyści. Dekadę później te krytyczne opinie wciąż są całkowicie uzasadnione.

Alternatywa ARIA — contentinfo

Także w tym przypadku, przynajmniej pod względem dostępności, przychodzi nam z pomocą specyfikacja ARIA. Punkt kontrolny `contentinfo` odpowiada treści standardowych stopek (czyli fragmentów umieszczanych u dołu stron WWW i zawierających kilka odnośników i jakieś inne informacje wyświetlane małą czcionką) pozbawionych efektownej nazwy. Specyfikacja ARIA opisuje elementy zawierające punkt kontrolny `contentinfo` w następujący sposób (*http://www.w3.org/TR/wai-aria/roles#contentinfo*):

> *Duży, zauważalny obszar zawierający informacje na temat nadrzędnego dokumentu.*
>
> *Przykładami informacji zamieszczanych w tym obszarze są prawa autorskie lub odnośniki do regulaminu.*

W odróżnieniu od elementu `<footer>` w dokumencie powinien znajdować się tylko jeden element, w którym została użyta rola `contentinfo`.

Rekomendacja

Elementu `<div role="contentinfo">` można użyć tylko raz, do stworzenia stopki strony. Można także dalej używać elementu `<div id="footer">` (bądź innego, dowolnie wybranego) z atrybutem `role="navigation"`, jeśli znajduje się w nim sporo ważnych elementów nawigacyjnych.

Kup książkę Poleć książkę

<main>

Jak przekonaliśmy się w rozdziale 3., trochę zajęło, nim element `<main>` został dodany do specyfikacji. Hickson uważał, że nie jest on niezbędny; prawdopodobnie dlatego, że reprezentował on jedyne faktycznie używane wcześniej rozwiązanie, które wszyscy chcieli utrwalić. Poza tym, element ten był całkowicie zrozumiały i sensowny dla przeciętnych projektantów i twórców stron. A Hickson sądził, że tak naprawdę musimy używać kodu tylko do reprezentacji tego, co *nie jest* treścią, a reszta, oczywiście, stanie się tą treścią z założenia.

Na szczęście, w tym przypadku przeważyła trzeźwość umysłu. W3C dodało element `<main>` do swojej specyfikacji języka HTML5.1, a następnie także do specyfikacji HTML5. Hickson zazdrośnie dodał go także do specyfikacji WHATWG, jednak postanowił go inaczej zdefiniować. Oto fakt: jedyny element, który powinien być prosty i zrozumiały, jest tym, co do którego nie udało się wypracować porozumienia.

Oto, w jaki sposób definiuje element `<main>` specyfikacja W3C (*http://www.w3.org/html/wg/drafts/html/master/grouping-content.html#the-main-element*):

> *Element `main` reprezentuje **główną treść** ciała dokumentu lub aplikacji. Główny obszar treści składa się z treści, która jest bezpośrednio związana z centralnym tematem dokumentu lub funkcjonalnością aplikacji lub ma za zadanie ich rozwijanie.*
>
> *Element `main` nie wyznacza sekcji i nie jest uwzględniany w planie dokumentu.*
>
> *Główny obszar treści dokumentu zawiera treści unikalne dla danego dokumentu i nie zawiera treści powtarzających się na innych stronach witryny, takich jak odnośniki nawigacyjne, informacje o prawach autorskich, logo witryny, banery reklamowe czy też formularze do wyszukiwania (chyba że taki formularz stanowi główną funkcjonalność dokumentu lub aplikacji).*
>
> *Zaleca się, by agenty zapewniające możliwość nawigowania przy użyciu klawiatury udostępniały sposób przejścia do elementu `main`, a kiedy to nastąpi, by zapewniały, że kolejnym elementem, do którego przejdzie użytkownik, będzie pierwszy element strony, umieszczony wewnątrz elementu `main`, który może zawierać miejsce wprowadzania. W ten sposób użytkownicy posługujący się klawiaturą zyskają prostą metodę pomijania całych bloków zawartości, takich jak elementy nawigacyjne.*
>
> *Autorzy nie mogą umieszczać w dokumencie więcej niż jednego elementu `main`.*
>
> *Autorzy nie mogą umieszczać elementu `main` wewnątrz elementów `article`, `aside`, `footer`, `header` oraz `nav`.*
>
> *Element `main` nie nadaje się do reprezentacji głównej zawartości podrzędnych sekcji dokumentu lub aplikacji. Najprostszym rozwiązaniem jest całkowita rezygnacja z oznaczania głównej treści sekcji podrzędnej i pozostawienie jej jako niejawnej; niemniej jednak autorzy mogą używać odpowiednich elementów grupujących oraz elementów sekcji.*
>
> *Autorom radzi się stosować atrybut ARIA `role="main"` w elemencie `main` aż do czasu, gdy w agentach zostanie zaimplementowane odpowiednie odwzorowywanie ról.*

To wydaje się całkiem proste, prawda? Element `<main>` reprezentujący główny obszar treści strony. Taki element może być tylko jeden. Nie bierze udziału w tym szaleństwie, jakim jest określanie planu dokumentu, i powinien zawierać atrybut ARIA `role="main"`. Brzmi super, prawda?

A oto definicja tego elementu zastosowana przez WHATWG:

> *Element `main` może zostać użyty jako pojemnik dla głównej treści innego elementu. Reprezentuje jego elementy podrzędne.*
>
> *Element `main` odróżnia się od elementów `section` oraz `article` tym, że nie jest uwzględniany w planie dokumentu.*

Właśnie tak — `<main>` „reprezentuje jego elementy podrzędne”. Trzeba też zwrócić uwagę na wszystko, co Hickson pominął: według specyfikacji WHATWG w dokumencie może się znaleźć kilka elementów `<main>` (co jest sprzeczne z definicją słowa *main* w języku angielskim), elementy `<main>` można zagnieżdżać wewnątrz

Kup książkę Poleć książkę

siebie („jesteś główną treścią głównej treści głównej treści!"), tak że w końcu nie będą już znaczyć nic; może za wyjątkiem tego, że „reprezentują czyjeś elementy podrzędne".

Ale przynajmniej specyfikacje zgadzają się odnośnie dwóch rzeczy: przede wszystkim, `<main>` nie bierze udziału w zamieszaniu z planem dokumentu, który sprawia, że pozostałe elementy strukturalne są tak zagmatwane; a po drugie, reprezentuje główny obszar treści. To miłe, jednak te różnice w standardach sprawiają, że HTML5 zaczyna w coraz większym stopniu przypominać podzielony, pełen kontrowersji świat rywalizujących ze sobą standardów i niezgodnych przeglądarek, od którego miał nas przecież uchronić.

Głównie bezużyteczna kontrowersja

Jak widać, istnieją pewne rozbieżności odnośnie sposobu, w jaki powinien być stosowany element `<main>`. Najlepsze w nich jest jednak to, że tak naprawdę nie mają one większego znaczenia, gdyż element ten jest nieprzydatny z tych samych powodów co pozostałe elementy sekcji — nie jest obsługiwany w starszych przeglądarkach i nie zapewnia dobrej dostępności.

Oprócz tego, w czasie pisania tej książki element `<main>` nie został jeszcze zaimplementowany w przeglądarkach Opera, Safari oraz Internet Explorer. Nawet jeśli dostępność czy też obsługa starszych przeglądarek nie znajduje się wysoko na naszej liście priorytetów, to jednak przeważająca większość twórców stron nie może z niego korzystać.

Trzeba pamiętać, że W3C oraz WHATWG najwyraźniej zawsze zgadzają się co do jednego: w ostatecznym rozrachunku to twórcy przeglądarek decydują o tym, co naprawdę będzie realne. Jak na razie element `<main>` należy raczej zaliczyć do fikcji.

Alternatywa ARIA — main

Na szczęście, istnieje jeszcze jeden punkt orientacyjny ARIA, którego możemy używać; jest nim po prostu `main`. Robi dokładnie to, na czym nam zależy, i służy tym, którym staramy się pomóc — osobom niewidomym.

Oto, w jaki sposób został on zdefiniowany w specyfikacji ARIA (*http://www.w3.org/TR/wai-aria/roles#main*):

> *Służy do oznaczania treści, która jest bezpośrednio związana z centralnym tematem dokumentu lub ten temat rozszerza. Rola `main` jest nieinwazyjną alternatywą dla odnośników „przejdź do głównej treści strony", w której opcja nawigacyjna pozwalająca na przejście do głównej treści strony (lub innego punktu orientacyjnego) jest zapewniana przez agenta przy wykorzystaniu okna dialogowego lub innych pomocniczych technologii.*

Rekomendacja

Sugeruję, by do tworzenia głównego obszaru treści dokumentu używać elementu `<div role="main">`.

Inne punkty orientacyjne ARIA

Oto kilka innych punktów orientacyjnych ARIA, które mogą się przydać podczas tworzenia kodu stron:

- `application`, służący do oznaczania programowych widżetów umieszczanych na stronie;
- `form` do oznaczania formularzy, za wyjątkiem tych do wyszukiwania, a to sugeruje kolejny punkt…
- `search` służący do oznaczania formularzy wyszukiwawczych umieszczonych na stronie.

Współdziałają one z innymi punktami orientacyjnymi, które zostały przedstawione w tym rozdziale:

- `banner`, służącym do oznaczania wszelkiego typu nagłówków;
- `navigation`, służącym do oznaczania — tak, odgadliście! — elementów nawigacyjnych;
- `complementary`, służącym do oznaczania pasków bocznych;

Kup książkę Poleć książkę

- `contentinfo`, służącym do oznaczania stopek;
- `main`, służącym do oznaczania głównej treści strony.

(Trzeba pamiętać, że punkty orientacyjne `banner`, `main` oraz `contentinfo` mogą się pojawić w dokumencie tylko jeden raz).

Więcej informacji na temat tych punktów orientacyjnych można znaleźć w doskonałym porównaniu ich z elementami HTML5 napisanym przez Steve'a Faulknera i dostępnym na stronie *http://blog.paciellogroup.com/2010/10/using-wai-aria-landmark-roles/*.

Jeszcze więcej ogólnych informacji na temat ARIA znajduje się w dokumentacji fundacji Mozilla, dostępnej na stronie *https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA*.

Stało się coś śmiesznego… Łagodna degradacja umarła, a JavaScript stał się obowiązkowy

W poprzednich rozdziałach kilkukrotnie wspominałem o problemach, których nowe elementy HTML5 mogą przysporzyć pewnej niewielkiej grupie użytkowników. Otóż znaczniki te mogą całkowicie zniszczyć układ stron niektórym spośród użytkowników przeglądarek Internet Explorer 6, 7 i 8, którzy wyłączyli obsługę języka JavaScript (ze względów osobistych bądź z powodu przesadnej troski o bezpieczeństwo zdarza się to częściej, niż można by sądzić).

Problem polega na tym, w jaki sposób przeglądarki Internet Explorer 6 – 8 obsługują „ogólne" elementy. Otóż dla tych przeglądarek elementem „ogólnym" jest każdy element, którego nie rozpoznają, niezależnie od tego, czy będzie to jakiś element wymyślony przez nas, taki jak `<mojsuperowyelement>`, czy też element `<nav>` HTML5. Zważywszy, że przeglądarki te w ogóle nie rozpoznają elementów ogólnych, nie jesteśmy w stanie określać ich wyglądu przy użyciu stylów bez zastosowania przy tym odrobiny kodu JavaScript, który powie przeglądarce, że elementy te istnieją.

To inteligentne skryptowe rozwiązanie zostało wymyślone przez Sjoerda Visschera i spopularyzowane przez Remy'ego Sharpa w 2009 roku (*http://html5doctor.com/how-to-get-html5-working-in-ie-and-firefox-2/*). Teraz, kiedy już możemy poinformować przeglądarki IE 6 – 8, że konkretne elementy *naprawdę* istnieją, możemy już bez przeszkód stosować style… możemy, prawda?

Nie, nie możemy.

Ale moglibyśmy?

Nie.

(Oficjalnie skrypt, o którym tu mowa, jest dostępny na stronie *http://code.google.com/p/html5shiv/*. Oprócz tego, trzeba poinstruować inne przeglądarki, by traktowały nowe elementy HTML5 jako elementy blokowe, o czym wspominałem w rozdziale 2.).

I teraz okazuje się, że trafiamy na wielką przeszkodzę, a konkretnie rzecz ujmując: użytkowników przeglądarek Internet Explorer 6 – 8 z wyłączoną obsługą języka JavaScript, którzy nagle nie są w stanie korzystać ze sprytnej poprawki do HTML5.

Ale chwila. Czy w obecnych czasach jest jeszcze ktoś, kto wyłącza obsługę JavaScriptu? A czy są dostępne jakieś dane, które mogłyby ułatwić nam znalezienie odpowiedzi na to pytanie?

Owszem, są.

Badania wykorzystania skryptów Yahoo

W 2010 roku firma Yahoo opublikowała wyniki badań stanowiące odpowiedź dokładnie na to pytanie — ilu użytkowników *ma* wyłączoną obsługę języka JavaScript? Okazało się, że było to 2,6 procenta osób odwiedzających amerykańskie witryny Yahoo (w tym znaczącym procentem tej wartości byli użytkownicy spoza USA), 1,29 procenta w Wielkiej Brytanii, 1,46 procenta we Francji i 1,28 w Hiszpanii. (Brazylia znalazła się na szarym końcu zestawienia z 0,26 procenta użytkowników).

A zatem, jeśli nie tworzymy witryny dla Brazylijczyków, to wciąż nie możemy pomijać osób, które mają wyłączoną obsługę języka JavaScript.

Kup książkę Poleć książkę

Więcej informacji na temat tych badań można znaleźć na stronie *https://developer.yahoo.com/blogs/ydn/many-users-javascript-disabled-14121.html*, natomiast zastosowana metodologia została opisana na blogu YDN prowadzonym przez Yahoo, a konkretnie na stronie *https://developer.yahoo.com/blogs/ydn/followup-many-users-javascript-disabled-16191.html*.

Załóżmy, że będzie to okrągłe 2 procent ruchu w przypadku głównych witryn w USA oraz że użytkownicy przeglądarek IE 6 – 8 stanowią 50 procent osób odwiedzających naszą witrynę. Oznacza to, że jedna na sto osób używa Internet Explorera 6 – 8 i ma wyłączoną obsługę języka JavaScript. A nawet gdyby to była tylko jedna osoba na *tysiąc*, oznaczałoby to, że dla witryny o umiarkowanym ruchu będzie to przynajmniej jedna osoba dziennie.

Co się zatem stanie z tymi osobami, jeśli zaczniemy używać nowych strukturalnych elementów HTML5?

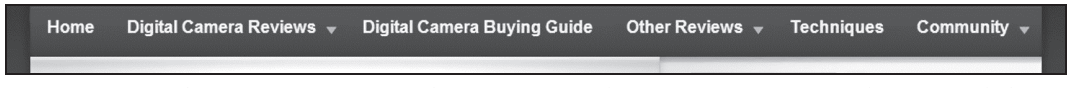
Jeśli ograniczymy się do użycia elementów `<section>` (bądź jakichkolwiek innych) umieszczanych wewnątrz innych elementów HTML i *nie będziemy* określać ich wyglądu przy użyciu stylów, to nie stanie się nic. Wciąż można tworzyć plan dokumentu w taki sposób, jeśli nam na tym zależy, o ile tylko będziemy pamiętać, że osoby niewidome poruszają się po witrynie, używając nagłówków.

Jeśli jednak użyjemy znaczników `<header>`, `<footer>`, `<nav>`, `<aside>` lub `<article>` (jak również `<section>`), w których zazwyczaj są stosowane style, to stanie się Coś Naprawdę Złego™.

Konkretnie rzecz biorąc, osoby używające przeglądarek, w których została wyłączona obsługa języka JavaScript, zobaczą witrynę o normalnym wyglądzie, *za wyjątkiem* tych jej fragmentów, w których zostały zastosowane elementy HTML5. Kiedy w tych elementach zostaną umieszczone wszystkie odnośniki nawigacyjne, nagłówki, paski boczne lub artykuły, to okaże się, że mamy poważne problemy. W tych nowych elementach HTML5, w odróżnieniu od pozostałej zawartości strony, nie zostaną zastosowane style CSS, więc na wiele różnych sposobów mogą zaszkodzić wyglądowi strony.

Oto, co się dzieje…

Rysunek 4.1 przedstawia przykładowy główny pasek nawigacyjny strony jednego z moich klientów, składający się głównie z elementu `<div id="nav">` zawierającego listę wypunktowaną `<ul>`.



Rysunek 4.1. *Pasek nawigacyjny witryny, w którym nie użyto elementów HTML5, wyświetlony w przeglądarce z włączoną obsługą języka JavaScript*

Z kolei rysunek 4.2 przedstawia ten sam pasek, utworzony przy użyciu elementu `<nav>` i wyświetlony w przeglądarce IE7 z wyłączoną obsługą języka JavaScript.

Rysunek 4.2. *Ten sam pasek nawigacyjny, w którym użyto elementu <nav> HTML5, wyświetlony w przeglądarce z wyłączoną obsługą języka JavaScript*

Niezbyt fajnie, prawda? A to tylko jeden element. Wyobraźmy sobie, co by się stało, gdyby nagłówek i stopka także popsuły się w podobny sposób, natomiast główna część strony, otoczenie treści oraz sama treść wciąż były prezentowane prawidłowo. Coś strasznego.

Co zrobić? A tak… XP

Najprostszym rozwiązaniem byłoby w ogóle zrezygnować ze stosowania stylów w przypadku wyświetlania strony w przeglądarkach IE 6 – 8 z wyłączoną obsługą języka JavaScript (co można zrobić, stosując instrukcje warunkowe oraz generując element `style` z poziomu JavaScriptu). Jednak osobiście nie polecam takiego rozwiązania — wciąż oznaczałoby to zupełnie niepotrzebne zniszczenie wrażeń osób używających tych przeglądarek.

Kup książkę Poleć książkę

A co z przeglądarkami IE9 i nowszymi? Na szczęście, rozpoznają one większość elementów HTML5, a oprócz tego poprawiono w nich sposób obsługi elementów ogólnych.

Może IE9 szybko zastąpi przeglądarki IE 6 – 8 i będziemy mogli zapomnieć o tych biednych użytkownikach, którzy wyłączyli w przeglądarkach obsługę języka JavaScript. Niestety, użytkownicy systemu Windows XP nigdy nie będą mogli skorzystać z przeglądarki IE9 — dla nich ostatnim przystankiem jest Internet Explorer 8. A zatem, dopóki będzie używany system Windows XP, wrażenia użytkowników korzystających z przeglądarki IE8 będą niepotrzebnie psute przez nowe elementy HTML5.

A nikt nie powinien tego znosić.

Elementy takie jak `<nav>` powinny (przynajmniej teoretycznie) pomagać osobom niedowidzącym. Jeśli jednak użyjemy tego elementu (lub innego) w sposób przedstawiony w specyfikacji i pokazywany przez propagatorów HTML5 — jako zamiennika istniejącego kodu strukturalnego — to nieumyślnie przysparzamy bardzo poważnych problemów innej mniejszości użytkowników.

Och, społeczności projektantów… co się stało?

Jednym z najbardziej niefortunnych aspektów wrzawy związanej z językiem HTML5 w kręgach projektantów stron WWW było to, jak szybko zapomnieliśmy o łagodnej degradacji (ang. *graceful degradation*). Oczywiście, nie chcemy być ograniczeni do najmniejszego wspólnego mianownika. Jednak przeglądarka IE8 jeszcze długo się nim nie stanie, a jest bardzo duża różnica pomiędzy udostępnianiem osobom, które wyłączyły w przeglądarce obsługę języka JavaScript, czegoś prostszego, a dawaniem im czegoś, co jest po prostu popsute i nie działa.

Chciałbym wyjaśnić, że nie stanowi dla mnie żadnego problemu fakt, że *Wasza witryna* może wymagać obsługi języka JavaScript (a w przeglądarkach IE 6 – 8 z wyłączoną obsługą tego języka będą w niej występować przeróżne problemy) — będzie to zapewne efekt Waszej świadomej decyzji. Niemniej jednak jest dla mnie problemem, że społeczność projektantów i twórców stron świadomie ignoruje tę kwestię i że nieświadomi projektanci i twórcy krzywdzą niewielką grupę użytkowników.

Już ta jedna sprawa całkowicie przekreśla jakąkolwiek przydatność nowych elementów, dlatego też sugeruję, by do tworzenia struktury strony wciąż używać elementów `<div>` uzupełnionych o odpowiednie role ARIA, zapewniające prawidłową dostępność. Role ARIA przynajmniej są w stanie poprawić dostępność witryny dla jednej grupy użytkowników, nie pogarszając jej dla innych.

Sam Hickson stwierdził (*http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2012-January/034506.html*):

> *Oczywiście, jeśli jesteście zadowoleni ze stosowania wszędzie* `<div class="...">`*, to nikt wam nie broni dalej tak robić.*

Wniosek — świętej pamięci strukturalne znaczniki HTML5

Być może najważniejszą informacją z tych wszystkich rozważań jest to, że nie jestem pewny, czy Ian Hickson — redaktor specyfikacji — sam dobrze wie, do czego mają służyć nowe elementy HTML5 lub do czego, według innych ekspertów, będą one używane.

Oto, co jeden z twórców przeglądarki Opera (oraz współautor książki *Introducing HTML5*, wydanej w 2010 roku przez wydawnictwo New Rider) Bruce Lawson napisał w 2009 roku na liście dyskusyjnej WHATWG (*http://lists.whatwg.org/htdig.cgi/whatwg-whatwg.org/2009-March/018888.html*, wytłuszczenie dodałem celowo):

> *W końcu nie znam żadnego programu, który by używał elementów* `time`*,* `section`*,* `footer`*,* `datagrid` *itd.,* ***jednak przeważnie oczekujemy, że wkrótce się pojawią****.*

A poniżej zamieszczam odpowiedź Hicksona:

> *A ja nie. Większość spośród nowych elementów ma jedynie ułatwiać stosowanie stylów, tak by nie trzeba było używać klas.*

Kup książkę Poleć książkę

I oto spontaniczne uzasadnienie Hicksona — *żeby nie trzeba było używać klas* — o którym wspominałem już wcześniej. Ciekawe jest jednak to, że Hickson nie oczekuje, by programy do czegoś tych elementów używały. Czyżby ograniczał się jedynie do podziału dokumentu na sekcje oraz tworzenia planu dokumentu?

To te elementy są zagubioną przyczyną. Gdyby projektanci *używali* ich jedynie zamiast klas (którymi już są), to nie zwracaliby uwagi na tworzone w ten sposób plany dokumentu. Mogliby przyjąć, że element `<header>` tworzy sekcję (a przecież tego nie robi), bądź też stosować element `<aside>` do tworzenia wyróżnionych cytatów (co spowodowałoby powstanie nowej sekcji); a to wszystko oznaczałoby bardzo dużo całkowicie nieprawidłowych planów dokumentów HTML5.

A zatem, skoro te nowe elementy będą jedynie wprowadzać jedno wielkie zamieszanie (choć przecież już to robią), to agenty użytkownika (a w szczególności czytniki ekranowe) będą miały bardzo małą motywację, by kiedykolwiek korzystać z planów dokumentów jako narzędzia nawigacyjnego — a właśnie tworzenie tych planów było podstawowym powodem wprowadzenia większości nowych elementów.

Czyż to nie jest smutne? Marzenie o wprowadzeniu podziału dokumentów HTML na sekcje, wyrażone w 1991 roku przez Tima Bernersa-Lee, doprowadziło do stworzenia specyfikacji HTML (po przerwanym projekcie XHTML 2.0), która dwie dekady później wprowadziła jedynie niezrozumienie i nadużycia.

A wszystko to jeszcze *zanim* w ogóle zaczęliśmy się zastanawiać nad nieścisłościami w specyfikacji, nad tym że elementy te przysparzają problemów ze stosowaniem stylów w przeglądarkach IE 6 – 8 oraz utrudniają tworzenie *kodu prostych dokumentów HTML*.

Poniżej przedstawiłem komentarz Johna Allsoppa, autora książki *Developing with Web Standards* (wydanej w 2009 roku przez wydawnictwo New Riders), dotyczący nowych elementów HTML5 i opublikowany na blogu Jeffreya Zeldmana (*http://www.zeldman.com/2009/07/13/html-5-nav-ambiguity-resolved/*):

> *Gdy zabrnąłem daleko w głąb specyfikacji, koncentrując się głównie na nowych, „semantycznych" elementach, takich jak `section`, `header`, `footer` i tak dalej, odkryłem wiele niejednoznaczności, słabo udokumentowanych cech, zdumiewających reguł zawierania [jak również] niemal bizantyjską złożoność.*
>
> *[…] wiele całkiem uzasadnionych zarzutów dotyczących XHTML 2.0 można także skierować pod adresem HTML5. Można także postawić wiele nowych, zwłaszcza dotyczących samej specyfikacji. Nadszedł czas, by je podać.*

Mamy jednak rozwiązanie określające, jak pisać kod naszych stron:

- Używać elementów `<div>` z nazwami klas (ewentualnie z unikalnymi wartościami atrybutu `id`, występującymi tylko jeden raz na danej stronie).
- Używać odpowiednich elementów nagłówków.
- Stosować odpowiednie punkty orientacyjne ARIA. To wystarczy.

W tak łatwy sposób można obecnie opisać strukturalny kod HTML.

Ta prostota została, niestety, gdzieś zagubiona w języku HTML5.

I to jest zła wiadomość. Dobrą wiadomością jest natomiast to, że jeszcze nie wszystko stracone. Możemy wykorzystać tę okazję, by używać nagłówków w lepszy, bardziej strukturalny sposób, nieznacznie ułatwiając w ten sposób życie osobom niewidomym i niedowidzącym. Z tego samego powodu warto zacząć stosować punkty orientacyjne ARIA. Jednocześnie możemy mieć radosną pewność, że nasze techniki tworzenia strukturalnego kodu HTML będą nam jeszcze mogły służyć przez wiele lat.

Chciałbym zakończyć ten rozdział stwierdzeniem, że pomimo całego krytycyzmu, z jakim odnoszę się do tej części specyfikacji HTML5 (oraz wsparcia, jakiego udziela jej środowisko projektantów i twórców stron WWW), to gdyby nie Ian Hickson oraz WHATWG, w ogóle nie *byłoby* języka „HTML5" w takiej postaci, w jakiej go dziś znamy. Być może wciąż czekalibyśmy, aż W3C pozbiera swoje dzieła w jedną całość. Gryzienie ręki, która nas karmi, nie jest moim celem — chciałbym się tylko nieco popodgryzać jej palce.

Teraz zajmiemy się słowem na *s* — semantyką.

Kup książkę Poleć książkę

Skorowidz

A

Kup książkę Poleć książkę

B

C

D

Kup książkę Poleć książkę

E

F

G

H

Kup książkę Poleć książkę

I

J

K

L

M

Kup książkę Poleć książkę

N

O

P

Q

R

S

Kup książkę Poleć książkę

T

U

V

W

Kup książkę Poleć książkę

X

Y

Z

Kup książkę Poleć książkę

Kup książkę Poleć książkę

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl

GRUPA WYDAWNICZA
Helion SA

TWÓRZ NOWOCZESNE WITRYNY WWW DZIĘKI HTML5!

HTML5

Wszystko, co powinniście wiedzieć o programowaniu. Przewodnik profesjonalisty

Dzięki nowym możliwościom HTML5 pozwala projektantom tworzyć lepsze, bardziej funkcjonalne strony WWW. Usługi geolokalizacyjne, przechowywanie danych w przeglądarce, zaawansowane wsparcie dla multimediów to tylko niektóre z nowości wprowadzonych w piątej wersji tego języka. Zastanawiasz się, jak wykorzystać ten potencjał?

Ta książka doskonale Ci to zademonstruje! Na początek poznasz krótką historię języka HTML i zobaczysz, z jakimi problemami jeszcze niedawno borykali się projektanci stron WWW. Następnie poznasz strukturę nowoczesnej strony WWW oraz elementy strukturalne HTML5. Niezwykle istotnym komponentem składni, pozwalającym na jeszcze lepsze indeksowanie witryn, są mikroformaty dostarczające cennych informacji wyszukiwarkom — rozdział piąty przedstawi Ci dogłębnie to zagadnienie. W dalszej części nauczysz się tworzyć atrakcyjne formularze oraz korzystać z potencjału nowego znacznika *<canvas>*. Na koniec poznasz format SVG oraz możliwości zastosowania HTML5 w rozwiązaniach mobilnych. Książka ta jest obowiązkową lekturą każdego projektanta stron WWW, który chce być na bieżąco z nowościami w sieci!

Dzięki tej książce:

- poznasz historię języka HTML
- właściwie zastosujesz elementy strukturalne HTML5
- zbudujesz funkcjonalny formularz
- poznasz format SVG i jego możliwości
- wykorzystasz potencjał HTML5 w Twoich projektach

Apress

sięgnij po WIĘCEJ
KOD KORZYŚCI

Nr katalogowy: 25693
Księgarnia internetowa: http://helion.pl
Zamówienia telefoniczne: 0 801 339900, 0 601 339900
helion.pl księgarnia internetowa
Sprawdź najnowsze promocje: http://helion.pl/promocje
Książki najchętniej czytane: http://helion.pl/bestsellery
Zamów informacje o nowościach: http://helion.pl/nowosci
Helion
Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

cena: 39,90 zł
ISBN 978-83-246-9422-8
9 788324 694228

Informatyka w najlepszym wydaniu