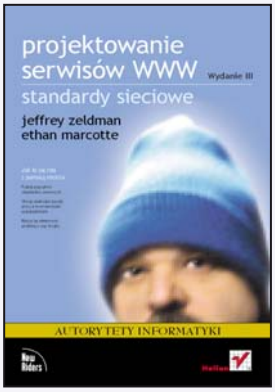
Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl


# Projektowanie serwisów WWW. Standardy sieciowe. Wydanie III




Autorzy: Jeffrey Zeldman, Ethan Marcotte
Tłumaczenie: Piotr Rajca
ISBN: 978-83-246-2658-8
Tytuł oryginału: Designing with Web Standards (3rd Edition)
Format: 158×235, stron: 448

**Jak to się robi z pomocą mistrza**

- Poznaj przyszłość standardów sieciowych
- Stosuj skuteczne zasady pracy z nowoczesnymi przeglądarkami
- Naucz się eliminować problemy i ciąć koszty

Przewodnik po standardach autorstwa Jeffreya Zeldmana ponownie kontratakuje! Trzecie wydanie udowadnia, że to wciąż najważniejsza książka dla wszystkich projektantów, którzy chcą tworzyć nowoczesne i funkcjonalne witryny – błyskawicznie się wczytujące, trafiające do masowego odbiorcy i tanie w utrzymaniu. Do nowej, zaktualizowanej wersji dodano niezwykle przydatne informacje o usprawnieniach, a także wyzwaniach, jakie stoją przed projektantami pracującymi w nieustannie zmieniającym się środowisku standardów sieciowych.

W tym wydaniu książki znajdziesz informacje na temat tego, jak za pomocą standardów rozwiązywać problemy powstające w wyniku stosowania starych sposobów projektowania witryn WWW. Dowiesz się także, jak stare i nowe standardy przekształcają internet w dynamiczną platformę do tworzenia solidnych, dostępnych aplikacji oraz pięknych i łatwych do odnalezienia treści. Przeczytasz też o tym, jak zapowiada się przyszłość standardów sieciowych.

- Prezentacja języków XHTML, HTML 5, CSS.
- Zasady tworzenia strukturalnego, semantycznego kodu.
- Realizacja solidnych układów, tworzonych w oparciu o CSS.
- Tworzenie nieinwazyjnego kodu JavaScript.
- Dodatkowe informacje dotyczące typografii i dostępności.
- Prezentacje kilku projektów, w których pokazujemy sztuczki i rozwiązania w zakresie stosowania standardów.

*W tym przemyśle Jeffrey Zeldman zajmuje miejsce gdzieś pomiędzy „Bogiem" a „guru" – i potrafi wykorzystać swą mądrość oraz dowcip, by opowiadać o tym, CZYM są standardy sieciowe, JAK należy je stosować oraz DLACZEGO powinniśmy zwrócić na nie uwagę.*

**Kelly Goto, autorka książki Web ReDesign 2.0: Workflow that Works**

*Czasami, bardzo rzadko, udaje się znaleźć autora, o którym myślimy sobie: „Ten gość jest inteligentny! I sprawia, że ja też czuję się mądrzejszy, bo w końcu zrozumiałem to zagadnienie".*

**Steve Krug, autor książek**
**„Nie każ mi myśleć! O życiowym podejściu do funkcjonalności stron internetowych"**
**oraz „Przetestuj ją sam! Steve Krug o funkcjonalności stron internetowych"**

# Spis treści

ROZDZIAŁ PIĄTY

# Nowoczesny układ znaczników

Część pierwsza nakreśliła w skrócie problemy biznesowe i produkcyjne będące wynikiem stosowania starych metod projektowania sieci, naszkicowała korzyści płynące ze stosowania standardów oraz odmalowała radosny obraz napędzanego standardami rozwoju sieci. W pozostałej części książki przejdziemy „od ogółu do szczegółu". Najlepszym sposobem, by zacząć, będzie poświęcenie kilku sekund na przeanalizowanie podstawowych zagadnień związanych z tworzeniem kodu stron, takich jak wybór wersji języka, jaką należy stosować, oraz sposoby kodowania niektórych, doskonale znanych, elementów stron: nagłówków, akapitów i list (podpowiedź: chodzi o to, jak to robić prawidłowo semantycznie).

Wielu projektantom i programistom myśl o ponownym analizowaniu kodu nie przypadnie do gustu. Z pewnością każdy, kto spędził na projektowaniu witryn więcej niż kilka tygodni, wie doskonale, o co chodzi w starym, dobrym HTML-u. Czy nie powinniśmy poświęcać naszego ograniczonego wolnego czasu, ucząc się nowszych, użyteczniejszych języków? Czy na przykład nie opłaca się bardziej studiować technologii działających po stronie serwera, takich jak PHP i jej podobne.

Odpowiedź brzmi: „Tak i nie". Technologie działające po stronie serwera mają istotne znaczenie przy tworzeniu dynamicznych witryn odpowiadających na zapytania użytkownika. Nawet tradycyjne witryny informacyjne mogą odnieść korzyści przez umieszczenie ich

treści w bazie danych i odwoływanie się do nich w miarę potrzeby przy użyciu PHP lub podobnych technologii. Niemal każda nowoczesna witryna korzysta z takich technologii, nawet skromny i niepozorny blog. Podobnie do standardów przedstawianych w tej książce, języki skryptowe wykonywane po stronie serwera oraz szkielety do tworzenia aplikacji, takie jak Ruby on Raili, CakePHP, Dijango czy Symfony, umożliwiają oddzielanie danych od interfejsu. Tak jak CSS, które niwelują konieczność umieszczania wszystkich fragmentów treści w pozbawionych semantycznego znaczenia komórkach tabel HTML, tak języki, na przykład PHP, i systemy zarządzania relacyjnymi bazami danych (jak choćby MySQL) pozwalają twórcom witryn uniknąć własnoręcznego pisania każdej ze stron.

### Co to takiego to PHP?

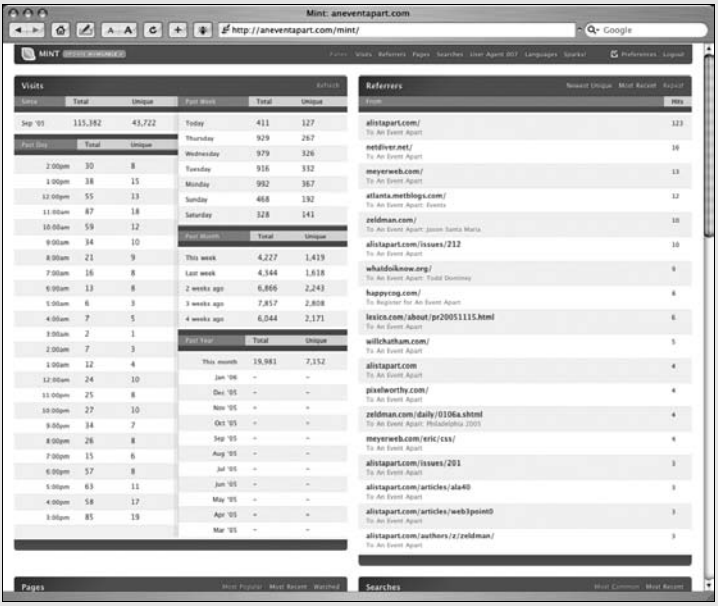
PHP (*www.php.net*) jest darmowym językiem skryptowym ogólnego przeznaczenia, który idealnie nadaje się do tworzenia stron internetowych i może być osadzany wewnątrz kodu HTML i XHTML. (Kod PHP można także umieszczać wewnątrz kodu HTML). Jego składnia przypomina języki, takie jak C, Java oraz Perl, i jest względnie łatwa do przyswojenia. PHP (skrót od *Hypertext Preprocessor*[1]) jest bardzo funkcjonalny, ale swoją popularność zyskał głównie dzięki możliwości współpracy z bazą danych MySQL (*www.mysql.com*). Cecha ta pozwala programistom i projektantom w łatwy sposób tworzyć dynamiczne strony i budować aplikacje sieciowe.

PHP stanowi projekt fundacji Apache (*www.apache.org*) i może być używany zupełnie za darmo, co stanowi jedno ze źródeł jego ogromnej popularności. Innym jest ogromny zestaw narzędzi do testowania i profilowania[2] kodu. Niezależni projektanci i programiści uwielbiają ten język (bądź uwielbiali, zanim nie poznali jego młodszego kuzyna — Ruby) i wykorzystują go nieustannie do tworzenia coraz to nowych stron i produktów (patrz rysunki 5.1 i 5.2). Z powodu jego skalowalności (nie wspominając o braku opłat) duże firmy, rzadko znane ze stosowania otwartych standardów, pokochały PHP. Przykładowo PHP napędza Yahoo.com już od 2002 roku (ostatnio jest przy tym używany otwarty szkielet PHP o nazwie Symfony [*http://www.symfony-project.org*]). Również IBM wsparł PHP swą potęgą, głównie za pośrednictwem formy Zend i szkieletu CakePHP (*http://www.internetnews.com/ent-news/article.php/3485806*).

[1] Mianem preprocesora określa się program wstępnie analizujący kod. Dla przykładu w języku C preprocesor uruchamiany jest przed właściwą kompilacją, aby dokonać odpowiednich podstawień w kodzie — *przyp. tłum.*

[2] Przez profilowanie należy rozumieć badanie wydajności kodu (jako całości lub poszczególnych jego fragmentów) — *przyp. tłum.*


**Rysunek 5.1.** Projektant i programista Shaun Inman użył PHP, MySQL, JavaScript i CSS do stworzenia Mint (havemint.com), rozszerzalnego narzędzia do raportowania potrafiącego odpowiedzieć, kto odwiedza i tworzy odnośniki do Twojej witryny, z jakiej przeglądarki korzysta i dużo więcej




**Rysunek 5.2.** Typowa instalacja Mint (widoczna tutaj pochodzi z witryny konferencji Event Apart)

### Co to takiego to PHP? — *ciąg dalszy*

PHP może współpracować z oprogramowaniem serwerowym Microsoftu, ale najczęściej używany jest w połączeniu z serwerem Apache. Apache pracuje zarówno na platformach Windows, jak i Unix (ze wskazaniem na te drugie). System operacyjny OS X firmy Apple ma wbudowany język PHP i serwer Apache, podobnie z resztą jak niemal wszystkie dystrybucje Linuksa.

PHP nie wymaga budowania interfejsu przy użyciu układu CSS i poprawnego semantycznego kodu znaczników, ale tam, gdzie trafiamy na oparte na standardach strony sieciowe, bardzo często okazuje się, że stoi za nimi PHP.

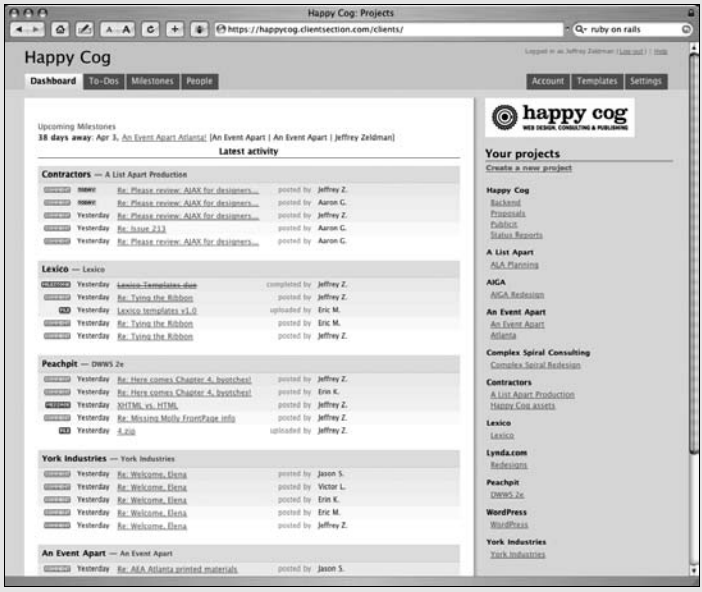
Jednak nawet dynamicznie wygenerowane strony staną się bezużyteczne, jeżeli będą niedostępne, niezgodne z wieloma przeglądarkami i urządzeniami lub zaśmiecone niepotrzebnym kodem znaczników. Jeżeli dynamiczna strona nie wyświetli się poprawnie w jakiejś przeglądarce lub jej załadowanie zajmie 60 sekund przez łącze modemowe, w sytuacji gdy wystarczyłoby w zupełności 10 sekund, technologie serwerowe na niewiele się zdadzą Twoim użytkownikom. Jeśli dodatkowo użytkownicy i wyszukiwarki nie będą w stanie znaleźć na witrynie interesujących ich informacji, gdyż zostały zagrzebane wewnątrz semantycznie bezsensownego kodu, wszystkie wysiłki włożone w utworzenie pięknego projektu, napisanie witryny i zapewnienie jej odpowiedniej obsługi ze strony mechanizmów działających po stronie serwera będzie można porównać do genialnych muzyków występujących przed pustą widownią.

### Czym jest Rails?

Ruby on Rails (*www.rubyonrails.org*) jest szkieletem do budowania aplikacji sieciowych dostępnym na licencji otwartego kodu, zoptymalizowanym pod kątem szybkiego, produktywnego, stabilnego programowania (patrz rysunek 5.3). Czym jest Ruby? Dobre pytanie. Ruby jest zorientowanym obiektowo językiem programowania stworzonym przez Yukihiro Matsumoto w 1995 roku i rozpowszechnianym jako wolne oprogramowanie na licencji open source. Łączy w sobie składnię inspirowaną językami Perl i Ada z funkcjami obiektowymi oraz posiada wspólne cechy z językami Lisp i Python

Zatem czym jest Rails? Rails jest szkieletem aplikacji działającym w oparciu o model projektowy MVC (ang. *model-view-controller — model-widok-kontroler*), napisanym w języku Ruby przez Davida Heinemeira Hanssona w lipcu 2004 roku. Rails został wyodrębnionym z aplikacji Basecamp firmy 37signals (patrz rysunek 5.4).


**Rysunek 5.3.** Ruby on Rails (www.rubyonrails.org) dostępny w postaci otwartych źródeł szkielet programistyczny. Jego podstawowe zasady to: „Nie powtarzaj się" oraz „Konwencja ponad konfigurację"



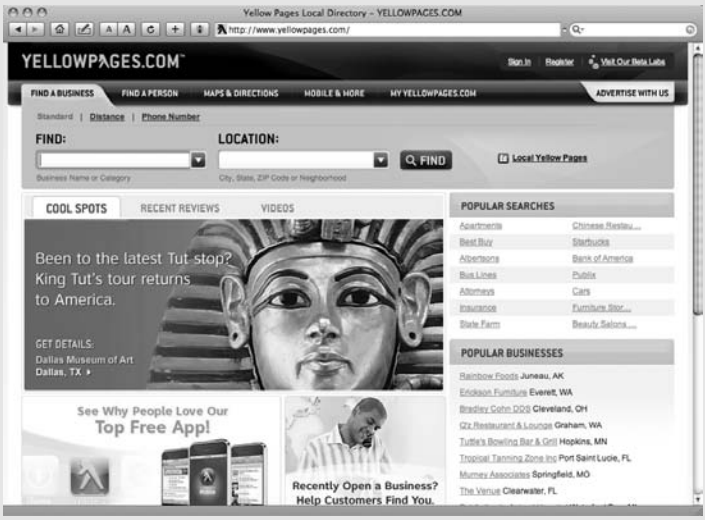

**Rysunek 5.4.** Basecamp firmy 37signals (basecamphq.com) to nieustające źródło prezentów. Nie tylko jest to wspaniała aplikacja do zarządzania projektami, ale również aplikacja, z której wyodrębniono Ruby on Rails

(Wersja 1.0 została opublikowana w postaci otwartego kodu źródłowego w grudniu 2005 roku). Programiści rzucili się na niego, ponieważ jego przewodnie zasady pozwalają szybciej pisać lepszy kod.

## Czym jest Rails? — *ciąg dalszy*

W większości środowisk programistycznych trzeba napisać tuzin wierszy kodu, aby przenieść zmienną na ekran (lub z ekranu). I to samo powtarza się za każdym razem podczas tworzenia nowego programu, dla każdej najmniejszej rzeczy, jaką robi ten program. To tak, jakby trzeba było pisać edytor tekstu zawsze wtedy, kiedy chcemy wysłać oficjalny list. Ruby on Rails odrzuca ten model. Wewnątrz Rails programiści nie muszą już kodować obsługi każdego najmniejszego szczegółu — muszą jedynie skonfigurować to, co jest niekonwencjonalne. Podobnie jak PHP, Ruby nie wymaga budowania interfejsu zgodnie ze standardami, ale te dwa elementy idą w parze dużo częściej niż osobno. Połączenie XHTML-a, CSS, JavaScriptu i Ruby on Rails zapewnia działanie wielu popularnych aplikacji internetowych, takich jak Twitter (patrz rysunek 5.5). Ruby on Rails jest także z powodzeniem wykorzystywany w tradycyjnych aplikacjach internetowych, na przykład yellowpages.com (patrz rysunek 5.6). Inne języki programowania — wśród nich PHP i Python — także udostępniają szkielety aplikacji MVC, podobne do Ruby on Rails.


**Rysunek 5.5.** Twitter (www.twitter.com), niezwykle popularna aplikacja do dystrybucji mikrowiadomości, powstała przy użyciu CSS, XHTML 1.0 Strict oraz Ruby on Rails




**Rysunek 5.6.** Poćwicz trochę palce, przeszukując książkę telefoniczną Yellow Pages (www.yellowpages.com), napisaną w XHTML 1.0 Transitional i korzystającą z Ruby on Rails

Krótko mówiąc, nie można rozdzielać obu technologii. Technologie działające po stronie serwera w połączeniu z bazami danych umożliwiają tworzenie sprytniejszych, bardziej funkcjonalnych stron, ale to, co dostarczają te strony, jest zawartością, która działa najlepiej, kiedy posiada czystą i prostą strukturę. I właśnie w tym miejscu większość z nas zawodzi (ale zawodzi także wiele systemów do zarządzania zawartością, na których polegamy).

## Ukryty schemat kiepskiego kodu

W trakcie pierwszej dekady istnienia przemysłu sieciowego projektowanie przypominało próbę nakarmienia mnóstwa wybrednych dzieci. Aby zbudować działającą stronę, posłusznie uczyliśmy się „diety" każdej przeglądarki. Obecnie wszystkie przeglądarki akceptują to samo pożywne jedzonko, a robią to od 2001 roku, ale wielu programistów jeszcze tego nie zrozumiało i wciąż dosypuje cukier do jajecznicy.

Złe jedzenie zatyka arterie, niszczy zęby i zmniejsza żywotność tych, którzy je spożywają. Zły kod znaczników jest szkodliwy dla krótkoterminowych potrzeb użytkowników oraz długoterminowego zdrowia zawartości stron. Do niedawna jednak fakt ten był przed nami ukrywany wskutek tolerowania przez popularne przeglądarki zaśmieconego i błędnego kodu, o czym mówiłem w rozdziale 1.

Tu oraz w kolejnych rozdziałach będziemy odkrywać na nowo zapomnianą naturę czystego, semantycznego układu znaczników oraz nauczymy się

### Inne platformy, inne obszary

Dwie inne platformy skryptowe używane do tworzenia dynamicznych stron i seksownych aplikacji sieciowych to Microsoft Active Server Pages .NET 2.0 (*www.asp.net*) oraz Adobe ColdFusion 8 (*www.adobe.com/software/coldfusion/*). Każda z nich ma swoje mocne strony oraz zagorzałe środowisko użytkowników. Java Serve Pages (JSP), jeszcze jedna technologia dynamiczna, jest powszechnie stosowana w ogromnych systemach dla przedsiębiorstw i zdecydowanie wykracza poza zakres tej książki.

Systemy do publikacji utworzone w niektórych z wymienionych języków mogą przy braku uwagi popsuć szablony stron przygotowane zgodnie ze standardami. W jednej z aplikacji, w tworzenie której zaangażowana była moja agencja, ASP (przed .NET 2.0) generował wszystko niepoprawnie, dopóki nie włączyliśmy do procesu HTML Tidy (*tidy.sourceforge.net*). Przypominało to obrzucanie błotem czystych ubrań, a następnie mycie ich wężem, ale pozwalało uzyskać poprawne strony z systemu, który był nieprzyjazny dla dobrego, czystego kodu. Pomijając już HTML Tidy, bardzo pomocne przy wprowadzaniu danych do systemów CMS może się okazać wykorzystanie takich narzędzi jak TinyMCE (*tinymce.moxicode.com*) lub WYMEditor (*www.wymeditor.org*). Istnieją także języki „quasi-znacznikowe", takie jak Textile Deana Allena (*www.redcloth.org/hobix.com*) lub Markdown Johna Brubera (*www.daringfireball.net/projects/markdown*), które także mogą pomóc osobom nieobeznanym z HTML-em w tworzeniu prawidłowego, zgodnego ze standardami kodu.

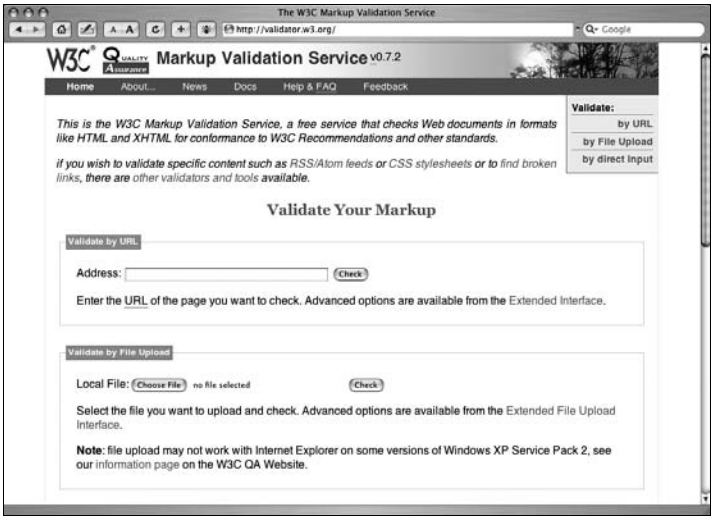
myśleć strukturalnie, zamiast postrzegać kod znaczników jako drugorzędne narzędzie projektowania. Jednocześnie przyjrzymy się XHTML-owi 1.0 — standardowemu językowi do projektowania witryn. Poznamy jego cele oraz korzyści wynikające z jego stosowania. Opracujemy strategię przejścia z HTML-a na XHTML. Przyjrzymy się także nowemu graczowi na rynku — językowi HTML 5 (*www.w3.org/TR/html5*).


### *SYNDROM BŁĘDNYCH ADRESÓW URL*

*Języki skryptowe często generują długie adresy URL zawierające niezakodowane znaki &, zastrzeżone w HTML/XHTML. W HTML-u i XHTML-u znak & używany jest do wskazania encji znakowej, takiej jak* `&#8217;`*, która oznacza poprawny typograficznie znak apostrofu w kodzie Unicode. Można to naprawić, stosując funkcję ColdFusion o nazwie*

*`URLEncodeFormat()`. ASP posiada podobną funkcję o nazwie HTMLEncode. Z kolei język PHP udostępnia funkcje* `urlencode()` *(*http://php.net/manual/pl/function.urlencode.php*),* `rawurlencode()` *(*http://php.net/manual/pl/function.rawurlencode.php*) oraz* `htmlentities()` *(*http://php.net/manual/pl/function.htmlentities.php*). We wszystkich tych przypadkach programiści mogą (i powinni) unikać problemu, przepuszczając wszystkie adresy URL przez wymienione funkcje przed umieszczeniem ich w kodzie strony.*

Dziwnym zbiegiem okoliczności jest to, że poprawne kodowanie z użyciem XHTML-a zachęca do pisania kodu w sposób strukturalny i zniechęca do tworzenia prezentacyjnych „sztuczek". W XHTML 1.0 Transitional takie sztuczki są uważane za przestarzałe, co oznacza, że możesz je stosować, jeśli musisz, ale jesteś *zachęcany* do osiągania tego samego rezultatu w inny sposób — na przykład przy użyciu CSS. W XHTML 1.0 i 1.1 Strict sztuczki prezentacyjnie są oficjalnie zabronione. Ich zastosowanie sprawi, że strona przepuszczona przez usługę weryfikującą poprawność kodu W3C (nazywaną walidatorem) nie przejdzie testu (patrz rysunek 5.7). (Jeżeli nie poznałeś jeszcze tego narzędzia, zrobisz to z pewnością, kiedy nauczysz się projektować i budować strony z użyciem standardów. Patrz ramka „Sprawdź to!").



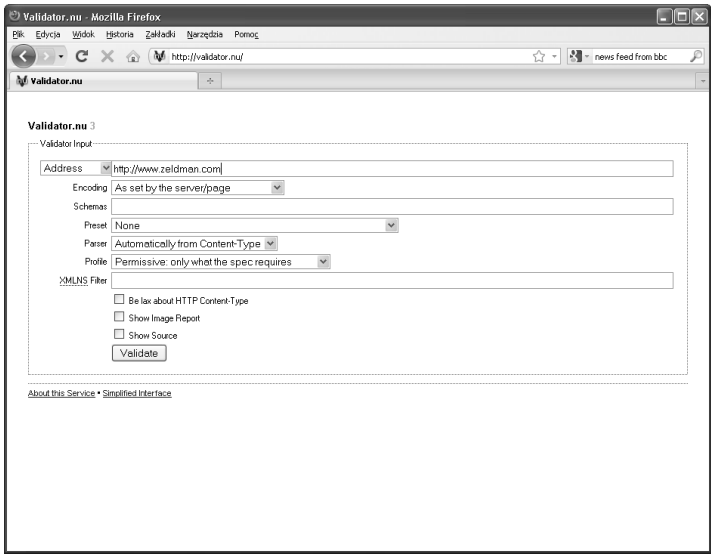

**Rysunek 5.7.** Projektanci i programiści korzystają z darmowej usługi sprawdzającej poprawność kodu oferowanej przez W3C (validator.w3c.org), aby się upewnić, czy ich strony są zgodne ze standardami

Niezależnie od tego, czy wybierzesz XHTML, czy Strict lub Transitional, doświadczysz uczącego pokory odkrycia, że „cała Twoja wiedza jest zła": łamanie wierszy (`br`) stosowane do symulowania listy, nagłówki wymuszające

### Sprawdź to!

Usługa weryfikowania poprawności stron (*validator.w3.org*) testuje strony zbudowane przy użyciu XHTML-a 4.01, XHTML-a 1.0 oraz XHTML-a 1.1 i ocenia ich zgodność ze specyfikacją języka. Usługa weryfikująca CSS (*jigsaw.w3.org/css-validator*) robi to samo w odniesieniu do arkuszy stylów. Grupa zajmująca się projektowaniem sieci w *htmlhelp.com* utrzymuje równie niezawodną usługę do sprawdzania układu znaczników (*www.htmlhelp.com/tools/validator*). Wszystkie trzy usługi oferowane są całkowicie za darmo.

**Rada:** HTML-owi maniacy — jeśli zajmujecie się już HTML-em 5, pamiętajcie, że nie wszystkie walidatory będą prawidłowo sprawdzać wasze strony. Jedną z usług, która to potrafi, jest walidator HTML 5 dostępny na stronie validator.nu (patrz rysunek 5.8). Także Validation Service W3C jest już w stanie sprawdzać poprawność kodu HTML 5. Cieszcie się i radujcie!




**Rysunek 5.8.** Być może ta strona nie jest szczególnie piękna, jednak validator.nu jest użytecznym narzędziem, które sprawdza poprawność kodu XML, HTML 5, itd. (www.validator.nu)

określony sposób wyświetlania, przezroczyste obrazki GIF do tworzenia białej przestrzeni... Będziesz zaszokowany faktem, że kiedyś stosowałeś tego typu sztuczki.

Zamiast korzystać z prezentacyjnych sztuczek, zaczniesz myśleć strukturalnie. Pozwolisz kodowi znaczników, by pełnił swoją rolę. Nawet w układach hybrydowych, stosujących tabele oraz inne elementy do prezentacji,

nauczysz się robić więcej przy użyciu CSS — na przykład usuwać złożone i nadmiarowe znaczniki koloru oraz atrybuty komórek tabel i zastępować je jedną lub dwoma regułami w globalnym arkuszu stylów. W miarę poznawania nowego języka kodowania będziesz zapominać stopniowo o złych przyzwyczajeniach. Przejdźmy zatem do rzeczy.

## Przeformułowanie czego?

Powołamy się na W3C i zacytujemy: „XHTML (*www.w3.org/TR/xhtml1*) jest przeformułowaniem HTML-a w XML". Mówiąc prościej i mniej precyzyjne, XHTML jest językiem znaczników bazującym na XML-u i działającym oraz wyglądającym jak HTML z kilkoma małymi, lecz znaczącymi różnicami. W przeglądarkach oraz innych klientach użytkownika XHTML 1.0 działa dokładnie tak samo jak HTML, chociaż niektóre nowoczesne przeglądarki mogą traktować ten język nieco odmiennie — o czym piszemy w następnym rozdziale. Z punktu widzenia projektantów i programistów, pisanie w XHTML-u 1.0 przypomina do złudzenia pisanie w języku HTML — tyle że z trochę bardziej ścisłymi regułami i jednym lub dwoma nowymi elementami, o których za chwilę.

W rozdziale 4. opisaliśmy XML (ang. *eXtensible Markup Language*) jako „superjęzyk", z którego programiści mogą wywodzić inne, dostosowane do własnych potrzeb języki znaczników. XHTML (ang. *eXtensible Hypertext Markup Language*) jest jednym z takich języków. XHTML 1.0 jest pierwszą i najbardziej zgodną wstecz wersją XHTML-a, stąd też najlepiej nadaje się do nauki i sprawia najmniej kłopotu starszym przeglądarkom.

Inne aplikacje i protokoły bazujące na XML-u liczone są w setkach, a ich popularność bazuje między innymi na zdolności do wymiany i transformowania danych przy minimalnym koszcie oraz zaledwie kilku (o ile w ogóle) kłopotach ze zgodnością — cnotach, jakie dzielą z XHTML-em. Pośród tych protokołów wymienić można Rich Site Summary (*blogs.law.harverd.edu/tech/rss*), Scalable Vector Graphics (*www.w3.org/TR/SVG*), Synchronized Multimedia Integration Language (*www.w3.org/TR/REC-smil*) i Resource Description Framework (*www.w3.org/RDF*). (Więcej informacji o tych językach można znaleźć w rozdziale 4.).

Każdy z tych protokołów pełni rolę w rozwijającej się sieci, ale żaden z nich nie jest tak istotny dla projektantów i programistów jak XHTML — i żaden z nich nie jest też równie prosty.

Po co w ogóle „przeformułowywać" HTML na XML lub cokolwiek innego? Z jednego powodu — XML jest językiem spójnym, czego nie można

powiedzieć o HTML-u. Jeżeli w XML-u otworzysz znacznik, musisz go zamknąć. W HTML-u niektóre znaczniki nigdy nie są zamykane, inne zawsze, jeszcze inne zależnie od woli programisty. Ta niekonsekwencja może spowodować praktyczne problemy. Przykładowo niektóre przeglądarki mogą odmówić wyświetlania strony HTML, która pozostawia niedomknięte komórki tabeli, mimo że specyfikacja HTML pozwala na taką praktykę. XHTML zmusza do zamykania wszystkich elementów, zatem unika się problemów z przeglądarkami i oszczędza czas niezbędny na testowanie oraz usuwanie usterek. Nie trzeba również pamiętać, które znaczniki należy zamykać, a które nie.

Co ważniejsze, jeżeli napiszesz stronę w języku bazującym na XML-u, będzie ona lepiej współdziałała z innymi językami, aplikacjami i protokołami opartymi na XML-u.

Skoro XML jest tak ważny, czemu nie napisać języka opartego na XML-u, który będzie działał dokładnie tak jak HTML? XML jest potężny i wszechobecny, nie można jednak zaserwować przeglądarce danych w surowej postaci XML-a i oczekiwać, że zrobi z nimi coś inteligentnego, na przykład wyświetli ładnie sformatowaną stronę internetową. Niestety, starsze przeglądarki nie poradzą sobie z wyświetleniem strony napisanej w XML-u. W istocie zatem XHTML jest technologią pośrednią, łączącą potęgę XML-a (w pewnej części) z prostotą języka HTML (w większości).

## Podsumowanie

Mówiąc ogólnie, XHTML to XML działający jak HTML w starych i nowych przeglądarkach oraz w większości urządzeń internetowych, poczynając od antycznych, takich jak Newton (produkowany w latach 90. ubiegłego wieku), poprzez urządzenia Palm, aż po iPhony. Praktyczna, przenośna i wydajna technologia.

XHTML jest równie prosty jak HTML — trochę prostszy dla początkujących, którzy nie posiadają złych przyzwyczajeń i być może ciut trudniejszy dla weteranów zajmujących się projektowaniem od samego początku lat 90. ubiegłego stulecia.

XHTML jest aktualnym standardem (zastępującym HTML 4), który ma na celu przywrócenie rygorystycznej, logicznej struktury i zawartości dokumentu zapewniającej lepsze współdziałanie z innymi standardami sieciowymi, takimi jak CSS i DOM, oraz dobrą współpracę z innymi istniejącymi i przyszłymi językami, aplikacjami oraz protokołami bazującymi na XML-u.

Lista wszystkich korzyści płynących ze stosowania XHTML-a została zamieszczona na końcu rozdziału.

### *WERSJA STRICT CZY TRANSITIONAL?*


*W pierwszych dwóch wydaniach tej książki zalecaliśmy stosowanie wersji XHTML 1.0 Transitional jako najbardziej wyrozumiałej spośród wszystkich wersji tego języka, najbardziej zbliżonej do tradycyjnych metod tworzenia stron oraz najłatwiejszej do poznania i zastosowania. Jeśli starasz się oduczyć starych nawyków lub chcesz uaktualnić istniejące witryny w łagodny i spokojny sposób, XHTML 1.0 Trasitional będzie najlepszym z możliwych wyborów.*

*Jednak z drugiej strony, aktualnie większość zwolenników przestrzegania standardów preferuje tworzenie stron w bardziej rygorystycznej wersji XHTML 1.0 Strict bądź też, jeśli są w stanie generować odpowiednie typy MIME dla wszystkich przeglądarek, z wyjątkiem Internet Explorera, w wersji XHTML 1.1 Strict. Wybór wersji Strict jako domyślnej jest częściowo kwestią mody — to sposób pokazania całemu światu (albo przynajmniej tej jego części, która zaprząta sobie głowę oglądaniem kodu źródłowego odwiedzanych stron), że nie uznajemy kompromisów, jeśli chodzi o stosowanie standardów. Zasadniczo to nic złego.*

## XHTML 2 — nie dla każdego

W momencie wydania pierwszej edycji tej książki wersja robocza specyfikacji XHTML 2.0 została przekazana do konsultacji społeczności projektantów i programistów. Kiedy zaczynaliśmy pracę nad trzecim wydaniem sześć lat później XHTML 2.0 jest nadal wersją roboczą (*www.w3.org/TR/xhtml2*) i nie został zaktualizowany od trzech lat. Mogłoby to świadczyć o tym, że świat nie dopomina się XHTML-a 2. I faktycznie, choć w jego specyfikacji można znaleźć kilka fascynujących rozwiązań, jednak nigdy nie znalazł wielkiego poklasku w społeczności programistów. 2 lipca 2009 roku W3C skróciło męki XHTML-a 2 (*http://www.w3.org/News/2009#item119*), zamykając prace nad nim, a społeczność osób zainteresowanych standardami oszalała z wściekłości (*http://www.zeldman.com/2009/07/07/in-defense-of-web-developers/*).

Głównym celem XHTML 2.0 było zbliżenie się do semantycznego ideału, nawet za cenę wyrzucenia na śmietnik dotychczasowych metod programowania. Początkowa wersja była faktycznie bardzo purystyczna. Z założenia XHTML 2.0 nie był zgodny wstecz z HTML lub XHTML 1.0. Porzucał znajome konwencje, takie jak element `img` (zastępowany przez element

`object`), znacznik `br` (zastępowany przez nowy element `line`, zamieniony później na `l`) oraz wiekowy znacznik zakotwiczenia; zamiast niego otrzymujemy do dyspozycji technologię zwaną `hlink`.

Programiści tak bardzo narzekali na tę ostatnią zmianę, że w późniejszej wersji element `a` został przywrócony. Ale w skorygowanym XHTML2 *każdy* element bądź grupa elementów strony mogły posiadać atrybut `href`. Należy zwrócić uwagę, że wielu programistom niezwykle spodobała się możliwość dodania atrybutu `href` do każdego elementu strony; rozwiązanie to zostało później wykorzystane w HTML-u 5, następcy XHTML-a 1.0, którym obecnie pasjonuje się sporo dzieciaków. Problem polega jednak na tym, że aktualnie większość przeglądarek obsługuje atrybuty `href` umieszczane jedynie w elementach `a`.

Jeśli chodzi o `img`, to w końcu udało mu się wrócić do XHTML 2 (*http://www.w3.org/TR/xhtml2/mod-image.html#sec_20.1*), został jednak uznany za przestarzały. Wciąż zamiast niego powinniśmy używać `object`, chociaż od dawna wiadomo, że `object` nie działa w Internet Explorerze z wyjątkiem wersji IE8. (Można to także wyrazić w bardziej uprzejmy sposób, mianowicie tak: to fajnie, że IE8 w końcu obsługuje element `object`).

Niektórzy projektanci przywitali proponowaną specyfikację XHTML 2 okrzykami radości. Inni narzekali, że wydaje się zbytnio bujać w obłokach, a za mało skupia się na faktycznych problemach tworzenia stron. (No dobrze, to chyba nam się wyrwało). Większość projektantów nie przywiązywała do niej wcale uwagi. Sześć lat później wszyscy, poza garstką nowatorów, nadal ją ignorują. Zamiast obsługiwać i rozwijać dwa standardy „języków znacznikowych przyszłości", spośród których tylko jeden interesował społeczność twórców stron WWW, W3C podjęło całkiem sensowną decyzję, by zakończyć prace nad XHTML 2.0.

Być może XHTML 2.0 jest już martwym pomysłem, jednak nie musimy się tym martwić. Żadna z przeglądarek nie przestanie bowiem obsługiwać XHTML-a 1. Podobnie, żadna z przeglądarek nie przestanie obsługiwać HTML-a 4. Witryny napisane poprawnie i zgodnie ze specyfikacją HTML 4.01 będą prawidłowo wyświetlane jeszcze przez długie lata. To samo dotyczy witryn napisanych prawidłowo i zgodnie ze specyfikacją XHTML 1.

Jednak po co używać XHTML-a, skoro w przyszłości językiem do tworzenia stron WWW ma być HTML 5? To całkiem zasadne pytanie, które twórcy stron używający HTML-a od ponad dekady niejednokrotnie sobie zadawali. Gdyby XHTML 5 miał się pojawić niebawem, a wszystkie przyszłe wersje przeglądarek miały obsługiwać wszystkie jego nowości (zaczynając

do elementów strukturalnych, takich jak `footer`, a kończąc na atrybutach `href` dodawanych do dowolnych elementów strony), to faktycznie zawracanie głowy nauką XHTML-a byłoby mało sensowne. Jedynym argumentem przemawiającym na korzyść XHTML-a mogłaby być nieco większa zgodność witryny z aplikacjami XML. Jednak prace nad standardem HTML 5 są jeszcze dalekie od zakończenia, a Internet Explorer (i w mniejszym stopniu także inne przeglądarki) nie obsługuje większości nowych elementów tego języka. A zatem obecnie wybór pomiędzy językami HTML i XHTML można sprowadzić do listy pięciu podstawowych zagadnień, przedstawionych poniżej.

## 5 najważniejszych powodów, dla których warto wybrać HTML

1. HTML działa prawidłowo we wszystkich przeglądarkach, a wszystkie przeglądarki (w tym IE) prawidłowo obsługują MIME HTML.
2. Choć najprawdopodobniej prace nad HTML 5 nie zostaną jeszcze zakończone przez kilka najbliższych lat, jednak najnowsze przeglądarki obsługują już niektóre jego elementy. Stwarza to doskonałą okazję, by już teraz rozpocząć naukę tej nowej, użytecznej wersji języka.
3. HTML traktuje błędy bardziej wyrozumiale niż XHTML.
4. HTML nie wymaga tak ścisłego zamykania elementów jak XHTML, a to z kolei może nieznacznie zmniejszyć zużywaną przepustowość. (A zużycie przepustowości przez `DOCTYPEE` HTML 5 jest najmniejsze z możliwych),
5. HTML 5 jest pierwszą wersją tego języka, zaprojektowaną pod kątem bogatych aplikacji internetowych, dlatego też wielkie firmy internetowe, takie jak Google, bez wątpienia będą nim bardzo zainteresowane. Jeśli zatem zajmujesz się aplikacjami internetowymi i odpowiada Ci kierunek rozwoju HTML-a 5, to teraz jest doskonały moment, by zacząć go poznawać i stosować.

Dodatkowym plusem jest fakt, że w żadnej z nowoczesnych przeglądarek obecność `DOCTYPE` HTML nie powoduje już automatycznego przechodzenia do trybu „dziwactw". Choć nie jest to żadną zaletą w porównaniu ze stosowaniem XHTML-a, jednak użycie HTML-a nie pogarsza już sytuacji programistów i nie zwiększa ryzyka przejścia przeglądarki do niebezpiecznego trybu „dziwactw". W dalszej części książki, w rozdziale 7., opiszemy podstawowe cele języka HTML 5, jego różnice w stosunku do XHTML-a oraz dokładniej przedstawimy jego elementy, reguły i składnię.

## 5 najważniejszych powodów, dla których warto wybrać XHTML 1

1. XHTML jest aktualnym standardem znaczników, zastępującym HTML 4.
2. XHTML jest zaprojektowany do współpracy z innymi językami skryptowymi, aplikacjami i protokołami bazującymi na XML. HTML nie posiada takiej możliwości.
3. XHTML jest bardziej spójny niż HTML, zatem mniej skłonny do stwarzania problemów związanych z funkcjonowaniem i wyświetlaniem treści.
4. Tworzenie w języku XHTML pozwala na wyzbycie się przyzwyczajenia do pisania prezentacyjnego kodu znaczników, a to z kolei może pomóc w uniknięciu problemów z dostępnością i niezgodnością przy wyświetlaniu stron w przeglądarkach różnych producentów. (Jeżeli piszesz strukturalny kod XHTML i umieszczasz wszystkie lub prawie wszystkie elementy prezentacji w CSS, czyli tam, gdzie powinny być, nie będziesz musiał dłużej martwić się o różnice pomiędzy przeglądarkami Firefox i Internet Explorer, takie jak puste komórki tabel, do których zastosowano atrybut szerokości).
5. Podobnie jak ćwiczenia z metronomem sprawią, że będziesz lepszym muzykiem, tak znaczenie, jakie w XHTML-u jest przykładane do prawidłowego formatowania kodu i przestrzegania reguł, stanowi doskonałą *platformę edukacji społecznej* dla wszystkich projektantów i programistów, którzy przez długie lata nawykli do tworzenia kodu HTML pozbawionego semantycznego sensu. Jeśli nawet za dwa lub trzy lata wrócisz do HTML-a 5, to dzięki poznaniu XHTML będziesz tworzyć bardziej przejrzysty i lepszy kod — nauczysz się bowiem przestrzegać semantyki wymuszanej przez ścisłe i rygorystycznie przestrzegane zasady.

## Podstawowy powód, dla którego nie warto wybierać XHTML-a

1. Nie znasz zasad XHTML-a.

Na szczęście, temu punktowi możemy zaradzić — patrz rozdział 6.