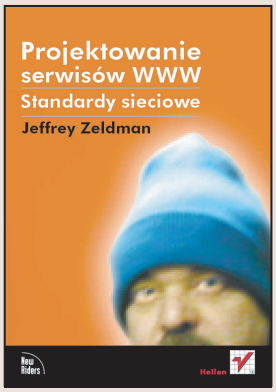
Projektowanie serwisów WWW. Standardy sieciowe



Autor: Jeffrey Zeldman
Tłumaczenie: Janusz Grabis (rozdz. 1–9),
Szymon Kobalczyk (rozdz. 10–16, dod. A)
ISBN: 83-7361-394-3
Tytuł oryginału: Designing with Web Standards
Format: B5, stron: 408

Tworzenie serwisów WWW może być bardzo frustrującym zajęciem. Jakże często spotykamy się z sytuacją, kiedy dopracowany i przetestowany w jednej przeglądarce serwis WWW wygląda koszmarnie w innej? Ile razy trzeba było przepisywać kod serwisu z powodu pojawienia się nowej przeglądarki i nowych „możliwości"? Najwyższy czas na zmianę. Standardy projektowania serwisów WWW opracowano właśnie po to, aby takie sytuacje nie miały miejsca. Mają one pomóc projektantowi w tworzeniu serwisów WWW, które będą wyglądać tak samo dobrze w każdej przeglądarce i na każdym urządzeniu.

Książka ta jest nieocenionym źródłem informacji dla każdego projektanta serwisów WWW. Zawarte w niej informacje umożliwią stworzenie serwisu, który nie tylko będzie uniwersalny, czytelny, łatwy w nawigacji, testowaniu i aktualizowaniu, ale także zgodny ze wszystkimi wytycznymi konsorcjum W3C dotyczącymi projektowania sieciowego.

Czytając tę książkę, dowiesz się:

- Dlaczego Twoje serwisy WWW wyglądają inaczej w różnych przeglądarkach
- Czym kierują się twórcy standardów sieciowych
- Jakie problemy wystąpiły przy pierwszych próbach znormalizowania metod projektowania
- Jak stworzyć czytelny i uniwersalny kod z wykorzystaniem języka XHTML
- W jaki sposób stworzyć strony WWW tak, aby były poprawnie wyświetlane na różnych urządzeniach
- Jak zdefiniować wygląd strony za pomocą stylów CSS
- Jak poradzić sobie z różnymi przeglądarkami
- Jak rozwiązać problemy związane z czcionkami na stronach WWW
- W jaki sposób zaimplementować w serwisie WWW mechanizmy ułatwień dostępu
- Jak wykorzystać na stronie skrypty oparte na modelu DOM (Document Object Model)

Dodatkowym, bardzo pomocnym materiałem, jest zestawienie informacji o używanych obecnie przeglądarkach internetowych.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 1.

99,9% witryn jest przestarzałych

Jest taka choroba, która dotyka niemal każdej witryny znajdującej się obecnie w sieci, od najskromniejszej strony domowej po portale korporacyjnych gigantów. Przebiegła i podstępna, rozprzestrzenia się niemal nierozpoznana, ponieważ bazuje na standardach przemysłu sieciowego. Chociaż projektanci i właściciele stron mogą o tym jeszcze nie wiedzieć, 99,9% witryn jest przestarzałych.

Strony mogą wyglądać i zachowywać się prawidłowo w popularnych przeglądarkach, których wersje są oznaczone cyfrą 4 lub 5. Ale poza tym tolerującym błędy środowiskiem zaczynają być już widoczne symptomy choroby i rozkładu.

W nowych wersjach przeglądarki Microsoft Internet Explorer, Opera Software, Netscape Navigator i Mozilla (przeglądarka typu open source bazująca na silniku Gecko, którego kod jest sterownikiem między innymi takich środowisk jak Navigator, CompuServe, AOL dla OS X, AOL China) starannie zbudowane układy stron zaczynają się rozpadać, a kosztowne mechanizmy zachowań przestają funkcjonować. Wraz z ewolucją tych przeglądarek wydajność stron będzie stale spadać.

W mniej znanych przeglądarkach, urządzeniach przystosowanych do potrzeb osób niepełnosprawnych, a także w zyskujących popularność palmtopach czy telefonach komórkowych z dostępem do sieci, większość stron nigdy nie działała, podczas gdy marginalna ich część działa doskonale. W zależności od potrzeb i budżetu właściciele witryn oraz sami projektanci ignorowali tego typu środowiska lub zauważali ich istnienie i zasilali je specjalnie przygotowanym kodem w taki sam sposób, jak dla zwykłych przeglądarek.

Aby zrozumieć bezsens takich działań i zobaczyć, w jaki sposób zwiększają one koszty i komplikują rozwój witryny, nigdy nie doprowadzając do osiągnięcia zamierzonego celu, musimy przeanalizować zachowanie nowoczesnych przeglądarek i dostrzec różnice występujące między nimi a ich starszymi niezgodnymi wersjami.

Nowoczesne przeglądarki i standardy sieciowe

Mówiąc w tej książce o „nowoczesnych" lub „zgodnych ze standardami" przeglądarkach, będziemy mieć na myśli przeglądarki, które rozpoznają oraz obsługują HTML i XHTML, kaskadowe arkusze stylów (CSS), ECMAScript oraz model obiektów dokumentu (W3C DOM). Wszystkie standardy zebrane razem pozwolą nam wznieść się ponad prezentacyjny układ znaczników, niezgodne języki skryptowe oraz będący ich wynikiem nieustający proces starzenia się witryn.

Do nowoczesnych przeglądarek zaliczają się między innymi: Mozilla 1.0+, Netscape Navigator 6+, Microsoft Internet Explorer 6+ dla Windows, Microsoft Internet Explorer 5+ dla komputerów Macintosh i nowsze oraz Opera 7. Porównanie najnowszych, zgodnych ze standardami przeglądarek zostało umieszczone w dodatku A książki „Nowoczesne przeglądarki: dobre, złe i okropne". Nie jest to lista wyczerpująca. Każda próba wyliczenia wszystkich przeglądarek zgodnych ze standardami jest z góry skazana na niepowodzenie. Mimo że będziemy stosować termin „zgodna ze standardami", proszę pamiętać o tym, co zostało powiedziane we wstępie: żadna z przeglądarek nie jest i nie może być *całkowicie* zgodna ze standardami.

Brak perfekcji przeglądarek nie zwalnia z dążenia do zgodności ze standardami. Miliony ludzi używają obecnie Internet Explorera 5 i 5.5 dla Windows. Jeśli chodzi o zachowanie standardów, te przeglądarki są gorsze od IE6/Windows, Netscape 6+ itd. Czy to oznacza, że jeśli nasz serwis odwiedzają użytkownicy tych przeglądarek, powinniśmy zapomnieć o standardach sieciowych? A może powinniśmy zaproponować im dokonanie uaktualnienia oprogramowania lub rezygnację z naszych usług? Nie. Projektowanie zorientowane na standardy sieciowe nie oznacza i nie wymaga „projektowania wyłącznie dla najnowszych przeglądarek". Użycie języka XHTML i stylów CSS nie jest równoznaczne z ignorowaniem użytkowników Netscape'a 4. Zaprojektowana i zbudowana zgodnie ze standardami strona najprawdopodobniej nie będzie wyświetlana dokładnie tak samo przez Netscape'a 4 i bardziej zgodne ze standardami przeglądarki. W zależności od przyjętej metody projektowej jej wygląd może być różny. Nie jest to jednak nic szczególnego. Wyjaśnimy to w drugiej części książki „Projektowanie i budowanie".

Nowy kod do nowej pracy

Nowoczesne przeglądarki nie są jedynie nowszymi wersjami tej samej starej serii. Różnią się zdecydowanie od swoich poprzedniczek. W wielu przypadkach zostały przebudowane od podstaw. Mozilla, Netscape 6/7 i przeglądarki bazujące na silniku Gecko nie są nowszymi wersjami Netscape Navigatora 4. IE5+/Mac nie jest uaktualnioną wersją IE4/Mac. Opera 7 nie bazuje na tym samym kodzie, który „napędzał" poprzednie wersje przeglądarki. Wszystkie zostały zbudowane na bazie nowego kodu w celu realizacji nowego zadania, a mianowicie zachowania jak największej zgodności ze standardami sieciowymi opisanymi w tej książce.

Przeglądarki lat dziewięćdziesiątych ubiegłego stulecia skupiały się natomiast na firmowych technologiach i nie zwracały zbytnio uwagi na standardy. Niektóre standardy były przez nie wręcz całkowicie ignorowane. Sytuacja taka nie była jednak traktowana jako poważne utrudnienie w procesie projektowania. Jeżeli, na przykład, przeglądarka nie obsługiwała standardu PNG (ang. *Portable Network Graphic*), projektanci nie używali obrazków w tym formacie. Kłopot polegał na tym, że stare przeglądarki wyświadczały „niedźwiedzią" przysługę standardom, oferując jedynie ich częściową obsługę, często niezgodną z założeniami. Byle jakie wsparcie dla tak podstawowych standardów jak HTML stworzyło niejednorodne środowisko publikacji, a w konsekwencji mało trwałe metody produkcji.

Kiedy pęka wyrostek robaczkowy u pacjenta, wykwalifikowani chirurdzy usuwają go całkowicie. Wyobraź sobie teraz, że zamiast nich zabieg wykonuje stażystka. Usuwa zaledwie połowę organu, przy okazji dźgając kilka sąsiednich i na końcu zapominając zaszyć pacjenta. Porównanie jest trochę makabryczne, ale doskonale oddaje podejście do obsługi standardów w starych przeglądarkach: niebezpiecznie niekompletne, nieudolne i ryzykowne dla zdrowia całej sieci.

Jeżeli Netscape 4 ignoruje reguły CSS zastosowane do znacznika `<body>` i dodaje losowe białe znaki do każdego elementu struktury na stronie, a IE4 traktuje ten znacznik prawidłowo, ale dla odmiany partaczy wyrównywanie, to którą ich wersję należy zastosować w projekcie? Niektórzy programiści w ogóle rezygnowali z CSS. Inni stosowali jeden arkusz stylów kompensujący błędy IE4 i drugi kompensujący gafy Netscape'a 4. Potrzebne było również stosowanie odmiennych stylów w zależności od tego, czy odwiedzający stronę jest użytkownikiem platformy Windows czy Macintosh (użytkownicy systemów Unix i Linux nie byli w ogóle brani pod uwagę).

Problemy z arkuszami stylów były zaledwie kroplą w morzu. Przeglądarki nie potrafiły jednakowo obsługiwać języka HTML, prezentować tabel lub interpretować języków skryptowych używanych do tworzenia interaktywnych elementów strony. Nie istniał jeden sposób budowania struktury zawartości strony. (Dokładnie mówiąc, istniał taki sposób, ale nie był obsługiwany przez żadną z przeglądarek.) Nie było żadnego ustalonego sposobu produkowania stron (tzn. istniał, ale nie był obsługiwany przez przeglądarki) lub dodawania zaawansowanych elementów do jej zawartości (także taki sposób istniał, lecz nie był rozpoznawany przez żadną ze starych przeglądarek).

Projektanci i programiści, walczący z ciągle pojawiającymi się niezgodnościami, wypracowali praktykę tworzenia wersji kodu dostosowanych do potrzeb każdej pojawiającej się przeglądarki. Było to wszystko, co w owym czasie mogliśmy zrobić, aby stworzyć witrynę dostępną dla więcej niż jednej przeglądarki lub systemu operacyjnego. Obecnie taka praktyka jest błędna, ponieważ nowoczesne przeglądarki obsługują te same otwarte standardy. Mimo to funkcjonuje nadal, pochłaniając zasoby, fragmentując sieć i generując niedostępne, mało użyteczne witryny.

Problem „wersji”

Tworzenie wielu wersji niestandardowego kodu (każdej dostosowanej do niestandardowych dziwactw tej lub innej przeglądarki) stanowi źródło ciągłego starzenia się stron — plagi dotykającej większość witryn. Trudno zwyciężyć w grze, której cele i zasady zmieniają się w trakcie meczu.

Mimo swojej kosztowności, bezsensowności i nietrwałości opisana praktyka nadal dominuje na rynku. Projektanci mający do czynienia z przeglądarką obsługującą standardy sieciowe traktują ją jak jedną z tych, które nie posiadają tej cechy. Tworzą kod, aby sprawdzić, czy jest to IE6, i „karmią” ją skryptami obsługiwanymi wyłącznie przez wytwory firmy Microsoft, chociaż IE6 radzi sobie ze standardami ECMAScript i DOM. Następnie czują się zmuszeni do napisania oddzielnych procedur dokonujących detekcji nowej przeglądarki Netscape, chociaż ta również potrafi obsłużyć wymienione standardy.

Jak sugeruje powyższy przykład, większość kodu podpatrującego wersje przeglądarek i urządzeń oraz generującego indywidualny kod jest niepotrzebna w obecnym klimacie tolerancji dla standardów. Nawet przy regularnych uaktualnieniach — na które niewielu właścicieli witryn może sobie pozwolić — skrypty dokonujące detekcji często zawodzą.

Na przykład przeglądarka Opera dla Windows identyfikuje się jako Internet Explorer. Robi to głównie po to, aby uniknąć blokowania przez witryny (w szczególności należące do sektora bankowego), które dokonują detekcji IE. Jednak skrypty napisane dla IE mają tendencję do „wysypywania” się w Operze. Kiedy zatem zidentyfikuje się ona jako IE (jest to domyślne ustawienie po zainstalowaniu) witrynie, której programista napisał kod specjalnie pod IE, liczba powstałych błędów oraz poziom frustracji użytkownika rośnie bardzo szybko. Mają oni możliwość zmiany ustawień w taki sposób, aby Opera identyfikowała się swoją prawdziwą tożsamością, zamiast podszywać się pod IE. O tej opcji wie jednak zaledwie garstka osób.

Oprócz skryptów dokonujących detekcji programiści piszą również rozbudowany kod prezentacji strony, który wymaga większej przepustowości od łącza klienta pragnącego ściągnąć stronę, jak i od udostępniającego ją serwera. Rozbudowany kod zmniejsza dostępność strony dla wyszukiwarek oraz niestandardowych przeglądarek i urządzeń internetowych. Stosowane strategie wywołują zatem często efekty, którym miały zapobiegać — niekonsekwentne prezentowanie witryn w różnych przeglądarkach (rysunek 1.1.).

Rozbicie witryny na różne wersje niesie ze sobą ciągle rosnące koszty oraz trudne do rozwiązania problemy. Witryny „DHTML” produkowane z uwzględnieniem firmowych specyfikacji Netscape’a 4.0 i Internet Explorera 4.0 nie działają w większości nowoczesnych przeglądarek. Czy właściciel takiej witryny powinien wydać jeszcze więcej pieniędzy na rozwiązanie tego problemu, zlecając programistom stworzenie piątej lub szóstej wersji strony? A jeśli nie ma na to pieniędzy? Wielu użytkowników zostanie zablokowanych.

Analogicznie projektanci mogą marnować wiele czasu i zasobów, tworząc „bezprzewodową” wersję swojej strony tylko po to, aby przekonać się, że zastosowany przez nich język znaczników jest przestarzały lub strona nie funkcjonuje w nowym urządzeniu

Rysunek 1.1. *Witryna MSN Game Zone (http://zone. msn.com/) obsługuje 7 arkuszy stylów, prezentuje się jednak niepoprawnie w najnowszych przeglądarkach. Chwali się 14 skryptami, wśród których jest bardzo opasły kod detekcji przeglądarek, ale nawet to jej nie pomaga. Jak widać, wykorzystanie dużej ilości kodu do rozwiązania problemu nie zawsze działa*

internetowym. Niektórzy w odpowiedzi na ten problem tworzą kolejną wersję witryny. Inni publikują żenujące komunikaty obiecujące wsparcie dla nowego urządzenia „w najbliższej przyszłości".

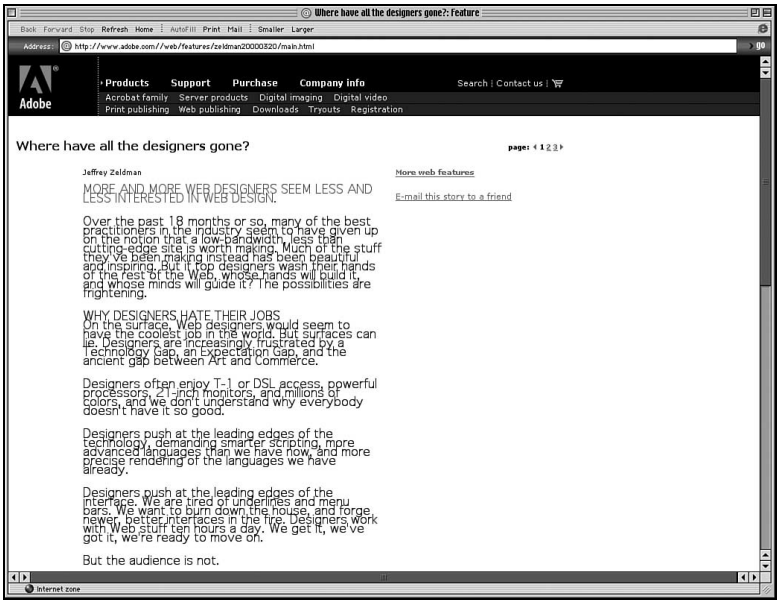
Nawet jeśli programista lub projektant zetknie się ze standardowymi technologiami sieciowymi, takimi jak XHTML i CSS, jego przyzwyczajenia pochodzące ze „starej" szkoły sprawiają, iż umyka mu sedno ich istnienia. Zamiast zastosować standardy do stworzenia jednej wersji, wielu programistów tworzy nadal co najmniej kilka wersji plików CSS zależnych od przeglądarki i/lub platformy, które niemal nigdy nie działają w oczekiwany sposób (rysunki 1.1 i 1.2).

Stosując tego typu praktyki, marnujemy czas i pieniądze, których zwykle nie mamy w nadmiarze, wręcz przeciwnie. Oliwy do ognia dolewa fakt, że mimo wysokich kosztów stosowane praktyki nie rozwiązują problemu. Strony zachowują się niekonsekwentnie, a użytkownicy mają do nich utrudniony dostęp.

Myślenie wsteczne

Zajrzyj do wnętrza jakiejkolwiek większej strony, poczynając od Amazon do Microsoft.com, od Sony po ZDNet. Zbadaj ich zawiły kod, osadzone kontrolki ActiveX i JavaScript (często zawierający źle działające skrypty rozpoznawania przeglądarek), a także z założenia źle użyte style CSS — o ile w ogóle je zastosowano. To cud, że te strony działają w jakiejkolwiek przeglądarce!

Działają, ponieważ pierwsze cztery lub pięć pokoleń przeglądarek Netscape Navigator oraz Internet Explorer toleruje jedynie specyficzny dla siebie kod. Taka sytuacja wręcz zachęcała do niechlujnego kodowania i tworzenia skryptów zależnych od producenta oprogramowania, aby zwyciężyć w rynkowej wojnie przeglądarek.



Rysunek 1.2. *Jedna ze stron firmy Adobe (http://www.adobe.com/) wygląda źle ze względu na nieprawidłowe formatowanie odstępów między wierszami (kolejne wiersze zachodzą na siebie). Zamiast stworzyć jeden lub dwa połączone arkusze stylów działające we wszystkich przeglądarkach, projektanci przedobrzyli, tworząc style zależne od przeglądarki i platformy o bardzo ubogim zestawie funkcji. Pozostając przy metodach „starej" szkoły, nie rozumiemy, że technologie CSS i XHTML niemal zawsze eliminują potrzebę tworzenia kilku wersji witryny*

Często niezgodne ze standardami witryny działają, ponieważ ich właściciele zainwestowali w drogie narzędzia do publikacji, które niwelują różnice między przeglądarkami przez generowanie wielu wersji kodu dostosowanego do danej przeglądarki lub platformy (patrz „Problem wersji"). Taka praktyka wystawia na próbę cierpliwość użytkownika korzystającego z modemowego dostępu do sieci przez zapychanie łącza rozwidlonym kodem, zagnieżdżonymi tabelami, generowanymi w locie obrazkami, a także przestarzałymi znacznikami i atrybutami.

Co to jest rozwidlanie kodu?

Kod stanowi podstawę każdego oprogramowania, systemu operacyjnego, generalnie mówiąc wszystkiego, co ma jakikolwiek związek z techniką cyfrową. Kiedy nad projektem pracuje więcej niż jedna grupa programistów, kod może „rozwidlić" się na kilka niezgodnych ze sobą wersji, szczególnie jeśli każda grupa próbuje rozwiązać inny problem lub przychylić się do ustaleń, o których inni nie słyszeli. Taki brak konsekwencji oraz sprawowania centralnej władzy nad kodem jest rzeczą złą.

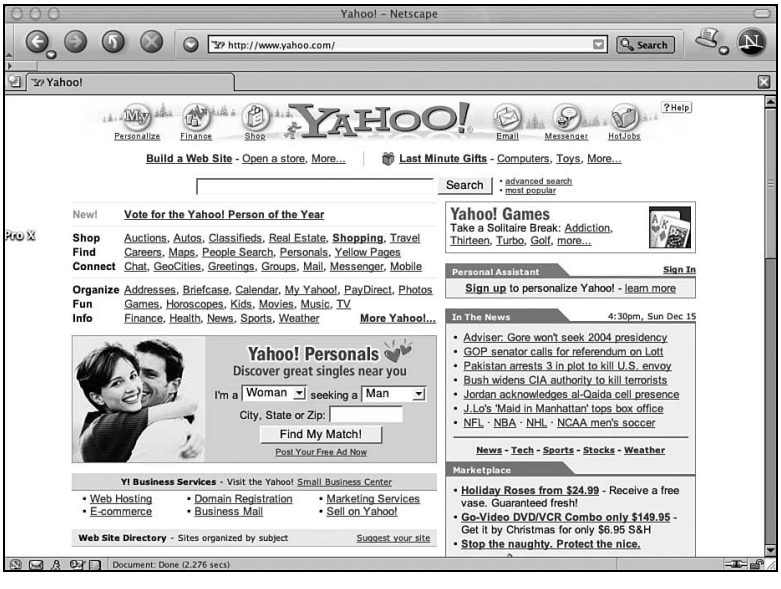
W tej książce termin „rozwidlanie kodu" oznacza praktykę polegającą na tworzeniu kilku różnych wersji kodu na potrzeby przeglądarek, które nie obsługują standardów ECMAScript i DOM (patrz „Problem wersji").

Kilka wersji kodu (który trzeba przesłać każdemu użytkownikowi) obciąża łącze właściciela witryny, podnosząc drastycznie koszty utrzymania serwisu. Im większa strona

i większy ruch na niej, tym więcej pieniędzy trzeba wydać na utrzymanie serwerów wykonujących zadania, których można uniknąć.

Liczby nie kłamią. Jeżeli strona zredukuje swój kod o 35%, zredukuje również o tyle samo koszty utrzymania łącza. Firma wydająca 10 000 zł rocznie mogłaby zaoszczędzić 3500 zł. Przy wydatkach rzędu 64 000 zł oszczędności wynoszą 22 400 zł.

Strona domowa Yahoo (rysunek 1.3) jest ładowana miliony razy dziennie. Każdy bajt zmarnowany na przestarzały kod przemnożony przez astronomiczną liczbę odsłon daje w wyniku gigabajty danych przesyłanych bez potrzeby. Można sobie tylko wyobrazić koszty ponoszone z tytułu takiego marnotrawstwa. Gdyby Yahoo zastąpiło jedynie niestosowane już znaczniki `<font>` (rysunek 1.4) przez wydajne style CSS, koszt ładowania każdej strony zmalałby wielokrotnie, a zyski firmy w konsekwencji wzrosłyby. Dlaczego zatem Yahoo nie wykonało takiego kroku?



Rysunek 1.3. *Strona domowa Yahoo (http://www.yahoo.com/)*

Tylko jedna odpowiedź wydaje się prawdziwa — firma pragnie, aby strona wyglądała dokładnie tak samo w starych przeglądarkach nieobsługujących CSS, jak i w nowych zachowujących zgodność z tym standardem. Ironia polega na tym, że nikt poza zarządem Yahoo nie przejmuje się jej wyglądem. Wiadomo bowiem, że olbrzymiego sukcesu witryna nie zawdzięcza wcale szacie graficznej (której nigdy nie posiadała), lecz oferowanym usługom.

Przykład tej, skądinąd interesującej, firmy (marnującej swoje łącza na dostarczanie wyglądu i zachowania[1], którego nikt tak naprawdę nie podziwia) mówi wszystko o zakorzenionym w umysłach projektantów podziwie dla „zgodności wstecz" i jej związku z użytecznością witryn oraz własnymi zyskami.

[1] Wszystkich interaktywnych cech witryny, jakie można stworzyć przy użyciu HTML-a i JavaScriptu — *przyp. tłum.*

Rysunek 1.4. *Yahoo od środka. Zobacz źródło, a przekonasz się, że kod służący do stworzenia tej prosto wyglądającej strony jest niewyobrażalnie skomplikowany*

```
<map name=m><area alt="My Yahoo!" coords="44,0,106,47" href=r/i1><area
alt=Finance coords="121,0,170,47" href=r/f1><area alt=Shop
coords="192,0,241,47" href=r/xm><area alt=Email coords="493,0,542,47"
href=r/m1><area alt=Messenger coords="558,0,618,47" href=r/p1><area
alt=HotJobs coords="634,0,683,47" href=r/hj><area alt=Help
coords="699,0,739,14" href=r/hw></map><img width=740 height=48 border=0
usemap="#m" src=http://us.i1.yimg.com/us.yimg.com/i/ww/m6v8x.gif
alt="Yahoo!"><table border=0 cellspacing=0 cellpadding=12><tr><td
align=center nowrap><font face=arial size=-1><table border=0
cellspacing=0 cellpadding=0><tr><td nowrap><font size=-1
face=arial><b><a href=s/46341>Build a Web Site</a></b> - <a
href=s/46342>Open a store</a>, <a
href=s/46343>More...</a></font></td><td align=center
width=21> </td><td bgcolor=cccccc width=1><spacer type=block
width=1 height=16></td><td align=center width=20> </td><td
width=18><img
src="http://us.i1.yimg.com/us.yimg.com/i/mntl/sh/16/giftbox.gif"
alt="Gift
Box" width=16 height=16 border=0></td><td nowrap><font size=-1
face=arial> <b><a href=s/46741>Last Minute Gifts</a></b> - <a
href=s/46345>Computers</a>, <a href=s/46346>Toys</a>, <a
href=s/46347>More...</a></font></td></tr></table></font></td></tr></table><tal
cellspacing=0 cellpadding=0 border=0><tr><td><form name=f
style="margin-bottom:0"
action="r/sx/*-http://search.yahoo.com/bin/search"><table cellpadding=0
cellspacing=0 border=0><tr><td><input type=text size=40
name=p> <input type=submit value=Search>  </td><td><font
face=arial size=-2>&#149; <a href=r/so>advanced search</a><br>&#149; <a
href=r/z1>most popular</a></font></td></tr></table></form><span
style="margin-top:-1em"></span></td></tr></table><table width=100%
cellpadding=0 cellspacing=0 border=0><tr><td
height=8></td></tr></table><table cellpadding=0 cellspacing=0
border=0><tr valign=top><td><table width=100% cellspacing=0
border=0><tr><td colspan=3><table width=100% cellpadding=0 cellspacing=0
border=0 bgcolor=eeeeee><tr><td height=1><table cellpadding=0
cellspacing=0 border=0><tr><td
height=1></td></tr></table></td></tr></table></td></tr><tr><td
colspan=3></td></tr><tr valign=top><td nowrap><font color=ff6600
face=arial size=-1><b>New!</b></font></td><td colspan=2><font face=arial
size=-1><a href=s/44801><b>Vote for the Yahoo! Person of the
Year</b></a></font></td></tr><tr><td colspan=3><table cellpadding=0
cellspacing=0 border=0><tr><td
height=2></td></tr></table></td></tr><tr><td colspan=3><table width=100%
cellpadding=0 cellspacing=0 border=0 bgcolor=eeeeee><tr><td
height=1><table cellpadding=0 cellspacing=0 border=0><tr><td
height=1></td></tr></table></td></tr></table></td></tr><tr><td
colspan=3></td></tr><tr valign=top><td nowrap><font face=arial
size=-1><b>Shop</b></font></td><td colspan=2><font face=arial size=-1>
```

Przestarzałe znaczniki: dodatkowy koszt dla właścicieli witryn

Załóżmy, że kod jednej strony zbudowanej według starych zasad zajmuje 60 kB. Zastąpienie znaczników `<font>` oraz innych przestarzałych znaczników czystym kodem z kilkoma regułami CSS zmniejsza rozmiar strony do 30 kB. (W praktyce możliwe jest zredukowanie 60-kilobajtowej strony do 22 kB lub nawet mniej, ale dla zachowania łatwości obliczeń przyjmijmy okrągłą liczbę, która reprezentuje oszczędności łącza internetowego rzędu 50%.) Rozważmy dwa typowe scenariusze przedstawione poniżej.

Redukcja łącza

Scenariusz: Samodzielnie utrzymywana witryna małego przedsiębiorstwa lub witryna należąca do sektora publicznego obsługuje ciągły strumień odwiedzających — kilkaset odsłon w danej chwili. Po zredukowaniu rozmiaru stron o połowę — poprzez konwersję znaczników prezentacji strony — do zwięzłego, czystego kodu XHTML firma oszczędza 1500 zł miesięcznie.

Jak to działa: Aby obsłużyć klientów przed konwersją, witryna potrzebowała dwóch linii T1 (1,544 Mb/s). Koszt dzierżawy każdej z nich wynosi 1500 zł na miesiąc. Po „ogoleniu" plików i zredukowaniu ich rozmiaru o 50%, firma dochodzi do wniosku, że jest w stanie obsłużyć tę samą liczbę klientów przez jedno łącze T1, tym samym redukując swoje koszty operacyjne o 1500 zł miesięcznie. Oprócz kosztów dzierżawy łącza zmniejszą się również nakłady na sprzęt komputerowy. Im prostszy jest kod

strony, tym szybciej jest ona dostarczana do użytkownika. Im szybciej jest dostarczana, tym mniej obciąża serwer — trzeba kupić, serwisować i modyfikować mniej serwerów. Jest to szczególnie istotne w przypadku serwerów, które muszą generować dynamiczną, sterowaną bazami danych zawartość — czyli zawartość, jaką posługują się niemal wszystkie współczesne witryny komercyjne.

Licznik megabajtów

Scenariusz: W miarę rozwoju komercyjnie hostowanej witryny jej właściciele dochodzą do wniosku, że każdego miesiąca płacą nieuzasadnioną karę za transfer plików, wynoszącą dziesiątki, a nawet setki złotych. Obcięcie rozmiaru plików o połowę sprowadza wysokość płaconych rachunków do przyzwoitego poziomu.

Jak to działa: Wiele firm oferujących usługi hostingowe przydziela swoim użytkownikom „wolny" od opłat miesięczny limit transferu plików — na przykład 3 GB. Poniżej tej wartości płacimy zwykłą stawkę miesięczną.Za przekroczenie limitu pobierane są dodatkowe opłaty, czasami *bardzo* duże.

W jednym „niesławnym" przypadku firma hostingowa Global Internet Solutions znokautowała niezależnego projektanta Ala Sacui szesnastoma tysiącami dolarów dodatkowych opłat, kiedy jego niekomercyjna strona, Nosepilot.com, przekroczyła dozwolony miesięczny limit transferu plików. Jest to przypadek ekstremalny, a klientowi ostatecznie udało się uniknąć zapłacenia kary dzięki udowodnieniu firmie zmiany warunków oferowanej usługi bez powiadomienia klientów (*http://thewebfairy.com/gisol/*). Kogo jednak stać na ryzyko płacenia niewyobrażalnych rachunków lub rozprawiania się z nieuczciwą firmą w sądzie?

Oczywiście nie każda firma hostingowa stosuje podobne praktyki. Pair.com na przykład obciąża klienta opłatą 1,5 centa za każdy megabajt ponad limit. Strona o małym ruchu sieciowym hostowana przez Pair.com może utrzymać się rocznie za kwotę 200 dolarów. Większe witryny, z większym ruchem na stronach oszczędzają najwięcej przez redukowanie rozmiaru plików. Niezależnie od tego, czy strona jest mała czy duża, odwiedzana przez miliony czy też przez garstkę ludzi, im mniejszy rozmiar plików, tym mniejszy ruch w sieci i mniejsze prawdopodobieństwo przekroczenia limitów. A już zupełnie na marginesie, najlepiej wybrać firmę, która stosuje nielimitowane transfery plików, zamiast karać swoich klientów za tworzenie popularnych stron.

Zgodność wstecz

Co programiści uważają za „zgodność wstecz"? Zapytani odpowiedzą: „zapewnienie obsługi wszystkim użytkownikom". I jak tu spierać się z takim argumentem?

W praktyce jednak „zgodność wstecz" oznacza stosowanie niestandardowych, zastrzeżonych (lub niepraktykowanych) znaczników oraz kodu, aby każdy użytkownik odwiedzający witrynę mógł doświadczyć tego samego, niezależnie od tego, czy używa IE2 czy Netscape 7. Zasada „zgodności wstecz" — traktowana jako święty Graal programowania — brzmi nieźle w teorii. Jednak jej koszt jest zbyt wysoki, a ona sama od zawsze opiera się na fałszywym założeniu.

Kod skondensowany a kod skompresowany

Gdy wygłosiłem wykład na temat standardów sieciowych, podszedł do mnie jeden z słuchaczy twierdząc, iż zyski wynikające ze stosowania czystego i dobrze ułożonego kodu nie są większe niż w przypadku stosowania kompresji kodu HTML.

Oprócz kondensowania kodu przez pisanie go w sposób przejrzysty i zwięzły (tzn. stosowanie struktur semantycznych zamiast przestarzałego formatowania z użyciem języka HTML), można również zwyczajnie skompresować kod w niektórych systemach serwerowych. Na przykład Apache oferuje moduł *mod_zip* kompresujący pliki HTML po stronie serwera. HTML jest ponownie rozpakowywany po stronie klienta.

Programista, z którym rozmawiałem, podał następujący przykład: jeżeli Amazon.com marnuje 40 kB na przestarzałe znaczniki oraz inne „śmieci", ale używa modułu *mod_zip* i kompresuje pliki do rozmiaru 20 kB, to nadmiarowy kod stron tej witryny nie generuje wydatków, o których mówiłem na wykładzie oraz w tej książce.

Jak się okazało, Amazon nie używa modułu *mod_zip*. W rzeczywistości narzędzie to jest rzadko używane w komercyjnych witrynach, przypuszczalnie ze względu na dodatkowe obciążenie związane z koniecznością kompresowania plików przed wysłaniem ich w świat. Wracając jednak do dyskusji z programistą, im mniejszy jest plik, tym lepiej zostanie skompresowany. Jeżeli oszczędzamy, kompresując 80-kilobajtowy pliku do rozmiaru 40 kB, wyobraźmy sobie, ile możemy zaoszczędzić, kompresując 40 kB do 20 kB. Oszczędności w pojedynczej sesji mogą wydawać się małe, ale ich wartość kumuluje się. Z czasem mogą znacznie zredukować koszty operacyjne i zapobiec innym wydatkom (na przykład dzierżawie dodatkowego łącza w celu zwiększenia przepustowości serwerów).

Oszczędności na łączu internetowym są tylko jedną z korzyści płynących z pisania czystego, dobrze ułożonego kodu, bardzo docenianą przez księgowych oraz klientów i równie prawdziwą dla tych, którzy stosują kompresję HTML-a.

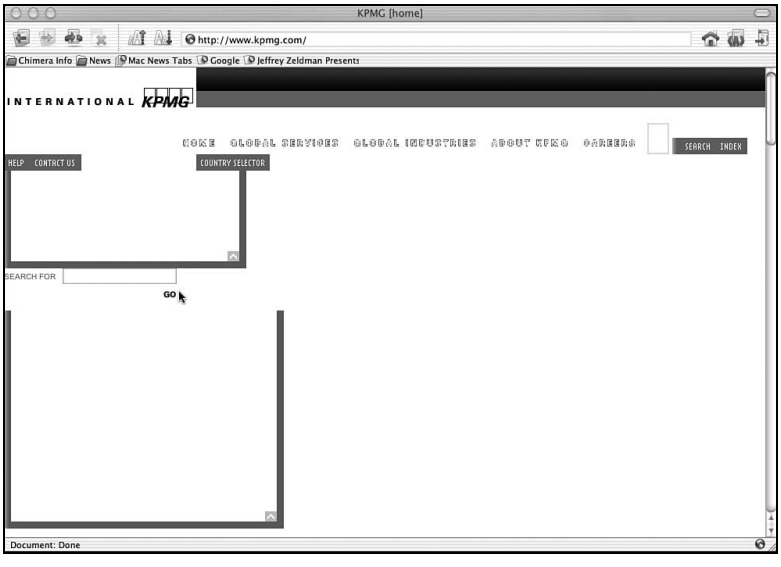
Nie istnieje prawdziwa zgodność wstecz. Zawsze istnieje punkt „odcięcia". Na przykład ani Mosaic (pierwsza przeglądarka graficzna), ani Netscape 1.0 nie obsługują układów opartych na tabelach HTML-owych. Zatem użytkownicy tych archaicznych przeglądarek nie mogą zobaczyć tego samego, co użytkownicy odrobinę nowszych narzędzi typu Netscape 1.1 lub MSIE 2.

Programiści i klienci głoszący ideę zgodności zmuszeni są do określenia „bazowej" przeglądarki, na przykład Netscape 3, i przyjęcia, że jest to najwcześniejsza przeglądarka, która obsługiwać będzie ich stronę (użytkownicy Netscape'a 2 nie mają szczęścia). Aby wypełnić zobowiązanie obsługi przeglądarki bazowej, wprowadzają do kodu szereg sztuczek, niestandardowych trików i okrężnych rozwiązań, które zwiększają ciężar każdej strony.

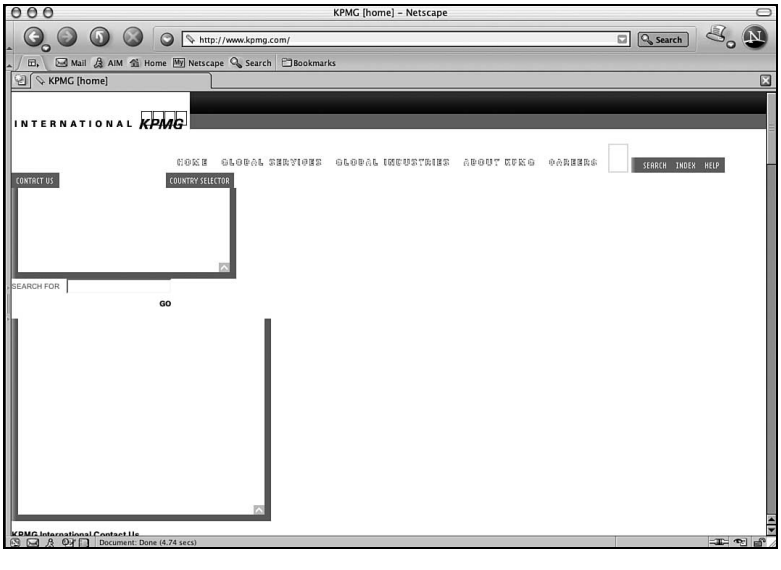
Jednocześnie piszą kilka skryptów, które rozpoznają typ przeglądarki i zasilają ją odpowiednim kodem. To dodatkowo zwiększa rozmiar stron, nakłada obciążenia na serwery i zapewnia nieustający proces starzenia się witryny aż do wyczerpania pieniędzy lub wypadnięcia z branży.

Blokowanie użytkowników nie wpływa dobrze na interesy

Podczas gdy niektóre firmy dokonują zamachu na swoje dochody, próbując zapewnić wsparcie nawet dla najstarszej przeglądarki, inne decydują się na obsługę wyłącznie jednej z nich. Ze względu na błędne założenia rośnie liczba stron projektowanych wyłącznie do współpracy z Internet Explorerem (czasem wyłącznie na platformie Windows), blokując tym samym 15 – 25% potencjalnych użytkowników i klientów (rysunki 1.5, 1.6, 1.7, 1.8, 1.9).

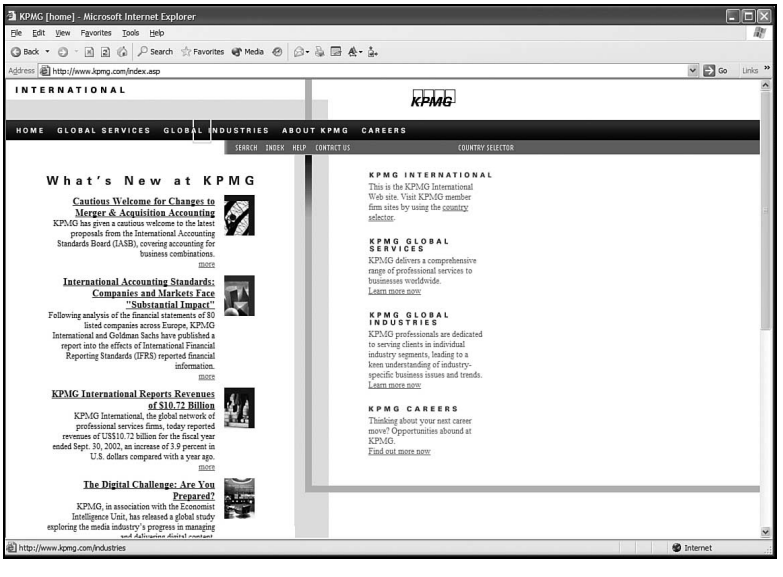


Rysunek 1.5. *Strona domowa KPMG (http://www.kpmg.com/) przeglądana w Navigatorze. Zupełna rozsypka układu jest efektem zastosowania kodu działającego wyłącznie w IE*



Rysunek 1.6. *Serwis KPMG jest równie bezużyteczny w Netscape'ie 7. Najwyraźniej firma nie przejmuje się tą grupą klientów*

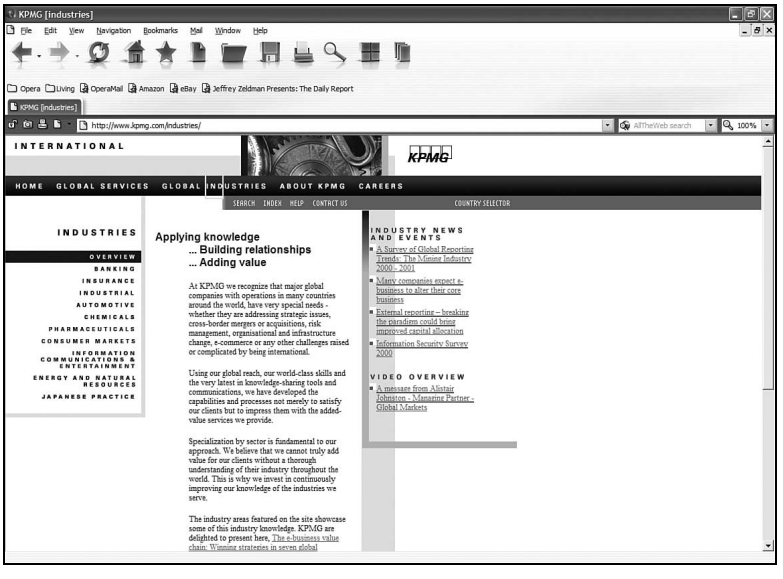
Rysunek 1.7. *Ta sam strona, tym razem w Internet Explorerze 5 dla komputerów Macintosh. Widok nie jest zachęcający (witryna działa poprawnie przez jakiś czas, a następnie zupełnie nieoczekiwanie „rozsypuje" się)*



Rysunek 1.8. *KPMG widziany przez IE6/Windows. Tutaj serwis w końcu działa*

Nie będziemy udawać, że rozumiemy podejście biznesowe firmy, która z założenia mówi NIE jednej czwartej swoich potencjalnych klientów. Tak duża liczba klientów stracona przez krótkowzroczne podejście nie powinna być akceptowana przez żadnego racjonalnego przedsiębiorcę lub instytucję publiczną z mandatem służenia społeczeństwu. Według statystyk sporządzanych przez NUA Internet Surveys (*http://www.nua.ie/surveys/*) ponad 650 milionów ludzi korzysta z internetu (wrzesień 2002). Sam policz, czy to się opłaca.

Powiedz, że nie przejmujesz się utratą 25% użytkowników, którzy pragną odwiedzić twoją stronę. Podejście „tylko IE" jest pozbawione sensu również dlatego, że nie istnieje



Rysunek 1.9. *Serwis poniekąd działa również w Operze 7 dla Windows, kiedy ta identyfikuje się jako IE (kiedy Opera identyfikuje się jako ona sama, serwis przestaje działać)*

gwarancja, że Internet Explorer (lub nawet przeglądarki dla komputerów stacjonarnych jako kategoria oprogramowania) będą dominować w przyszłości.

Klika lat przed powstaniem tej książki Netscape Navigator cieszył się większym powodzeniem na rynku niż Internet Explorer obecnie. W owym czasie funkcjonował pogląd, iż Netscape jest jedyną liczącą się przeglądarką, również wśród programistów tworzących strony internetowe. Niezliczone miliony dolarów później rynek uległ zmianie. Strony działające wyłącznie „pod Netscape'em" zostały wyrzucone na śmietnik obok internetowej superautostrady, po której miały się poruszać.

Czy podobny los spotka witryny „wyłącznie dla IE"? Kłopot polega na tym, że nie ma odpowiedzi na to pytanie. Jedyną pewną rzeczą w sieci jest zmiana. Odpowiedź twierdzącą sugerują jednak ciągle rozwijające się urządzenia korzystające z internetu, a także pogląd, iż projektowanie z przeznaczeniem wyłącznie dla jednej przeglądarki „stacjonarnej", kosztem wszystkich innych, jest drogą donikąd.

Poza tym, co pokaże niniejsza książka, standardy umożliwiają projektowanie dla wszystkich przeglądarek i urządzeń z łatwością i szybkością, z jaką robi się to obecnie dla jednej z nich. Gdzieś pomiędzy nakręcającą koszty zgodnością wstecz a krótkowzrocznością polegającą na budowaniu dla jednej przeglądarki znajduje się jedyne słuszne rozwiązanie — projektowanie z użyciem standardów sieciowych.

Zarówno technika tworzenia wielu wersji witryny, jak i jawnie podejmowanie decyzji obsługi wyłącznie jednej przeglądarki nie pomogą dzisiejszym witrynom funkcjonować w świecie przyszłego oprogramowania oraz rozwijać się w ciągle ewoluującym świecie urządzeń mobilnych. Jeżeli obecne metody będą kontynuowane, koszty oraz złożoność witryn będzie wzrastać do momentu, kiedy na ich tworzenie stać będzie wyłącznie największe firmy.

W naszych wysiłkach oferowania jednakowego wyglądu i zachowania w środowisku niezgodnych ze standardami przeglądarek — chcemy tworzyć witryny wyglądające jak magazyny drukowane na papierze i zachowujące się jak oprogramowanie — straciliśmy z oczu prawdziwy potencjał sieci jako bogatego i wielowarstwowego medium dostępnego dla wszystkich.

Zgubiliśmy go, kiedy projektanci i programiści, walczący o sprostanie wymaganiom produkcyjnym podczas bumu internetowego, nauczyli się niestandardowych metod tworzenia witryn zorientowanych na jeden wybrany produkt, w efekcie wprowadzając nas do obecnej ery, którą można określić erą „niezgodności".

Na szczęście okres „niezgodności" w rozwoju sieci kończy się w chwili, kiedy czytasz te słowa, zabierając ze sobą niezliczone witryny. Jeżeli jesteś właścicielem, zarządzasz, projektujesz lub budujesz strony, ten dzwon bije również dla ciebie.

Droga do Pacanowa

Na początku 1997 roku powszechną praktyką było pisanie w języku JavaScript dla przeglądarek Netscape i JScript (języku podobnym do JavaScriptu) dla przeglądarek Microsoft. Równie powszechne było stosowanie JavaScriptu (obsługiwanego wyłącznie przez Netscape'a) i ActiveX (dostępnego wyłącznie dla IE/Windows) do wysyłania przeglądarkom potrzebnego im kodu. Tak postępowaliśmy z przeglądarkami w wersji 3.0.

Praktyki takie nie przysługiwały mniej znanym programom, takim jak Opera czy choćby Internet Explorer dla komputerów Macintosh, ale zadawalały „większość" użytkowników sieci i dzięki temu szybko urosły do rangi normy branżowej. Jeżeli chcieliśmy tworzyć aktywne strony, które oferowały coś więcej ponad ładny wygląd, nie mieliśmy innego wyboru, jak tylko przestrzegać ustalonych procedur.

Pod koniec roku 1997 wprowadzono na rynek przeglądarki Netscape i Microsoft w wersji 4.0, zapewniające potężne możliwości dynamicznego języka HTML (DHTML), które, jak łatwo można zgadnąć, były zupełnie ze sobą niezgodne. Ponadto były również niezgodne ze swoimi poprzednimi wersjami (to, co działało w Netscape'ie 4, nie działało w Netscape'ie 3), nie wspominając już o zupełnym braku zgodności z mało znanymi przeglądarkami, które pokornie obsługiwały podstawowe standardy jak HTML zamiast tworzenia swoich własnych języków i atrybutów.

Czy taka sytuacja była normalna? Netscape i Microsoft sądziły, że tak, podobnie jak wielu programistów i projektantów. Pozostali, niezgadzający się z tą sytuacją, nie mieli alternatywy, musieli zagryźć zęby i tworzyć kilka wersji witryny, aby zapewnić jej „profesjonalizm".

Na ile różnych sposobów należało kodować

Był DHTML dla Netscape'a 4. Następnie niezgodny DHTML dla Internet Explorera 4, który działał niemal wyłącznie w środowiskach Windows. Do tego dochodziły nie-DHTML-owy JavaScript dla Netscape'a 3 i nie-DHTML-owy kod dla IE3. W ostateczności pod uwagę należało jeszcze brać inne wersje kodu przeznaczone dla mniej

popularnych przeglądarek. Nawet najmniej interesująca strona potrzebowała zatem minimum kilku rozwidleń kodu.

Niektórzy projektanci ograniczali się do dwóch wersji (jednej przeznaczonej dla IE4 i drugiej dla Netscape'a 4.0), a wymagający użytkownicy mieli do wyboru zaopatrzyć się w czwartą wersję przeglądarki albo zapomnieć o korzystaniu ze strony. Pozostali, z jeszcze mniejszymi budżetami, nastawiali się na obsługę tylko jednej przeglądarki i generalnie przegrywali.

Projekt standardów sieciowych, który wystartował tuż po pojawieniu się na rynku przeglądarek w wersji 4., ocenił, iż potrzeba pisania czterech lub więcej niezgodnych wersji każdej funkcji zwiększa koszt projektowania i produkowania witryny o minimum 25% — koszt ponoszony przez klienta.

Środowisko programistów odpowiedziało na tę ocenę wzruszeniem ramion. Sieć była bardo gorącym towarem, a klienci chętnie płacili bardzo wysokie rachunki, dlaczego zatem duże agencje interaktywne miały martwić się wysokimi kosztami wynikającymi z konieczności tworzenia kilku wersji kodu. W końcu jednak bańka mydlana prysła, budżety zaczęły maleć lub zamrażać się, agencje zwalniały swoje obroty lub zupełnie wypadały z rynku. Nagle nie było już nikogo, kogo stać było na trwonienie pieniędzy w taki sposób.

Kiedy rynek zmieniał się w następstwie zwolnień i bankructw, pojawiła się nowa generacja przeglądarek obsługujących stworzony przez W3C standard DOM. Co oznaczało to posunięcie? Możliwość wyrzucenia na śmietnik kilku wersji witryny i stworzenia projektu w oparciu o nowy standard, który w końcu pojawił się w zasięgu ręki. Jak myślący ekonomicznie przemysł odpowiedział na tę długo oczekiwaną wiadomość? Kontynuował pisanie rozwidlonych wersji kodu, tworzył witryny przeznaczone wyłącznie dla IE/Windows lub przechodził na technologię Flash firmy Macromedia. Jak na biznes kreowany przez wizje przemysł sieciowy potrafi być nadzwyczajnie krótkowzroczny.

Dobre traktowanie złego kodu

Na początku zdobywania wiedzy z zakresu programowania adept sztuki przechodzi przez fazę „wkładania śmieci i otrzymywania śmieci". Języki takie jak C i Java nie zachęcają do stosowania poprawnych konstrukcji, one tego wymagają.

Podobnie pierwszą rzeczą, jakiej uczy się grafik komputerowy, jest to, że jakość materiałów źródłowych determinuje jakość produktu końcowego. Zacznij od dobrej jakości fotografii wykonanej w dużej rozdzielczości, a otrzymasz porządny wydruk lub obrazek na stronę WWW. Zacznij od niskiej jakości fotki lub wykonanego w małej rozdzielczości zdjęcia cyfrowego, a otrzymasz wynik, na który nie warto będzie nawet spojrzeć. Z wysokiej jakości rysunku można wyprodukować dobre logo dla witryny, ale operacja odwrotna jest niemożliwa. Wkładasz śmieci, otrzymujesz śmieci.

Jednak tradycyjne przeglądarki nie działają w taki sposób. Ich swoboda w przyjmowaniu błędnych danych posunięta jest do absurdu. Akceptują wszystko, poczynając

od źle skonstruowanych znaczników, przez niedziałające odnośniki, aż po błędny kod JavaScript. W większości przypadków, mimo takich błędów, pokazują stronę tak, jakby była skonstruowana poprawnie. Ta swoboda od zawsze zachęcała czołowych projektantów i programistów do wyrabiania w sobie złych przyzwyczajeń, których sami mogli być nawet nieświadomi. Jednocześnie przekonywała pozostających bardziej w cieniu fachowców z branży do postrzegania technologii takich jak XHTML, CSS i JavaScript jako pogardliwie prymitywnych.

Przy braku szacunku dla narzędzia trudno stosować go poprawnie. Rozważmy poniższy fragment kodu zaczerpnięty z drogiej komercyjnej strony należącej do firmy współzawodniczącej na wymagającym rynku, przedrukowany tutaj znak w znak:

```
<td width="100%"><ont face="verdena,helvetica,arial" size="+1"
↳color="#CCCC66"><span class="header"><b>Join now!</b></span></ont></td>
```

Nonsensowny znacznik <ont> jest źle zapisanym znacznikiem <font>. Dzięki zastosowaniu bardzo wydajnego narzędzia do publikowania witryn ta literówka przewija się tysiące razy przez kod strony. Jeśli pominiemy sam błąd, przedstawiony fragment kodu powinien wyglądać znajomo. Być może jest on wręcz odzwierciedleniem kodu, który stosujesz na swoich stronach. W kontekście tej konkretnej witryny całość można było zredukować do następującej postaci:

```
<h3>Join now!</h3>
```

Połączony z odpowiednią regułą w arkuszu stylów ten prościutki, strukturalny fragment kodu będzie robił dokładnie to samo, co nieefektywny, niestandardowy i błędny kod przedstawiony wyżej. Mniejsza ilość kodu wpłynie pozytywnie na wydajność łącza serwera i klienta, a także ułatwi przejście do bardziej elastycznej formy strony wspomaganej przez znaczniki XML. Ta sama komercyjna strona zawiera również błędny skrypt w języku JavaScript:

```
<script language=JavaScript1.1src="http://foo.com/
↳Params.richmedia=yes&etc"></script>
```

Oprócz innych usterek pozbawiony cudzysłowów atrybut określający język styka się z atrybutem wskazującym źródło skryptu. Innymi słowy, przeglądarka została poproszona o użycie nieistniejącego języka skryptowego (JavaScript1.1src).

Jakby na to nie patrzeć, strona powinna przestać działać, informując projektantów o popełnionym przez nich błędzie i motywując do jego natychmiastowego usunięcia. Mimo to jeszcze do niedawna strona była serwowana najpopularniejszym przeglądarkom w niezmienionej postaci. Jest to tylko jeden z niezliczonych przykładów zupełnego braku szacunku dla procesu projektowania interfejsu witryny przez doświadczonych programistów.

Śmietnik utrzymywany w kodzie może być niebezpieczny dla zdrowia witryny

W miarę jak nowe przeglądarki zaczynają przestrzegać standardów sieciowych, stawiają coraz bardziej rygorystyczne wymagania projektantom i programistom. Ich tolerancja błędnego kodu oraz odnośników jest coraz mniejsza. Znaczenia zaczyna nabierać zasada

wkładania śmieci i otrzymywania śmieci — znajomość standardów sieciowych staje się niezbędnym elementem wiedzy dla każdego, kto projektuje lub buduje witryny.

Wyrządzoną szkodę można naprawić. Możemy projektować i budować witryny w taki sposób, aby działały na różnych przeglądarkach i platformach, zapobiegając jednocześnie problemom związanym z brakiem zgodności i blokowaniem użytkowników. W ten sposób zbudujemy pomost do potężniejszej, bardziej dostępnej i racjonalnie zaprojektowanej sieci.

Lek na przestarzałość można znaleźć w zbiorze powszechnie wspieranych technologii nazywanych „standardami sieciowymi". Przez naukę projektowania i budowania z użyciem standardów sieciowych możemy zagwarantować zgodność w przód każdej witrynie, jaka wyjdzie spod naszej ręki.

„Napisz raz, publikuj wszędzie", obietnica standardów sieciowych jest czymś więcej niż tylko pobożnym życzeniem. Standardy sieciowe są wśród nas już dziś, dzięki zastosowaniu metod, jakie odkryjemy w tej książce. Mimo że obecnie najpopularniejsze na rynku przeglądarki ostatecznie skłoniły się ku obsłudze standardów sieciowych, wiadomość ta nie dotarła jeszcze do każdego pracującego w branży projektanta i programisty. Nowo powstające strony nadal zawierają niestandardowe znaczniki oraz kod. Ta książka ma odmienić ten stan.

Lek

Po długiej batalii projektantów i programistów z producentami najpopularniejszych na rynku przeglądarek możemy w końcu skorzystać z technik, które gwarantują jednakowy wygląd i zachowanie stron we wszystkich przeglądarkach.

Stworzone przez członków konsorcjum W3C i inne organizacje zajmujące się ustanawianiem standardów oraz obsługiwane przez współczesne przeglądarki firm Netscape, Microsoft, Opera i innych, technologie takie jak CSS, XHTML, ECMAScript (standardowa wersja JavaScript) i W3C DOM pozwalają projektantom:

- bardziej precyzyjne kontrolować układ, położenie i typografię w przeglądarkach graficznych z możliwością zmiany prezentacji strony według potrzeb użytkownika,
- projektować zaawansowane funkcje interaktywne witryn działające w różnych przeglądarkach i na różnych platformach,
- przestrzegać praw i wytycznych dotyczących funkcji ułatwiania nawigacji bez pogarszania wyglądu, wydajności lub stopnia skomplikowania,
- zmienić projekt witryny w ciągu godzin, a nie tygodni, i tym samy zredukować koszty pracy,
- obsługiwać wiele przeglądarek bez konieczności tworzenia wielu wersji witryny i z bardzo małą ilością lub brakiem rozwidlonego kodu,

- obsługiwać niestandardowe urządzenia, poczynając od bezprzewodowych gadżetów i telefonów komórkowych z dostępem do sieci, aż po czytniki kodu Braille'a oraz czytniki zawartości ekranu stosowane przez osoby niepełnosprawne — ponownie bez konieczności ponoszenia dodatkowych kosztów i tworzenia wielu wersji witryny,
- oferować wersje stron „do wydruku", często bez konieczności tworzenia oddzielnych stron przeznaczonych specjalnie do wydruku lub bazowania na drogich systemach do tworzenia dokumentacji drukowanej,
- oddzielać styl od struktury i zachowania, dostarczać kreatywne układy stron wsparte przez rygorystyczną strukturę dokumentu umożliwiającą reprodukowanie dokumentów sieciowych w zaawansowanych środowiskach publikacji,
- przejść z HTML-a, języka „starej" sieci, na układ znaczników bazujący na niezwykle potężnym standardzie, jakim jest XML,
- zapewnić, iż dobrze zaprojektowane i zbudowane strony będą działać doskonale w przeglądarkach zgodnych ze standardami i zachowywać się poprawnie w starszych programach,
- zapewnić, iż dobrze zaprojektowane i zbudowane witryny będą działać w nowych przeglądarkach i urządzeniach, łącznie z tymi, których jeszcze nie zbudowano lub nawet nie wymyślono. Jest to obietnica zgodności w przód,
- … i więcej, co pokaże niniejsza książka.

Zanim będziemy mogli nauczyć się, w jaki sposób standardy umożliwiają realizację wymienionych wyżej obietnic, musimy przyjrzeć się metodom „starej" szkoły, aby dowiedzieć się, w jaki sposób generują one nieustający cykl starzenia się witryn. Wyjaśni to rozdział 2. „Projektowanie i budowanie z użyciem standardów".