

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Projektowanie serwisów WWW. Standardy sieciowe. Wydanie II

Autor: Jeffrey Zeldman

Tłumaczenie: Szymon Kobalczyk

ISBN: 83-246-0774-9

Tytuł oryginału: [Designing with Web Standards \(2nd Edition\)](#)

Format: B5, stron: 464



Dostosuj swoje witryny do obowiązujących standardów

- Poznaj standardowe technologie sieciowe
- Naucz się pisać strony poprawnie wyświetlane we wszystkich przeglądarkach
- Zredukuj koszty utrzymania witryny

Dzięki ludziom takim jak Jeffrey Zeldman w świecie technologii internetowych coraz większą uwagę przywiązuje się do standardów. Dotyczy to także producentów popularnych przeglądarek internetowych, dlatego wreszcie możliwe jest tworzenie efektywnych witryn, które wyglądają identycznie u użytkowników korzystających z różnych programów. Jednak nie jest to jedyna zaleta stosowania standardów. Za ich pomocą możesz sprawić, że Twoje strony będą działały szybciej, a ich aktualizacja stanie się dużo łatwiejsza, co przełoży się również na koszty utrzymania witryny.

„Projektowanie serwisów WWW. Standardy sieciowe. Wydanie II” to zaktualizowana wersja niezwykle popularnego, praktycznego przewodnika po świecie standardowych technologii internetowych. Dowiesz się z niego, czym są standardy sieciowe oraz dlaczego warto się do nich stosować. Poznasz sposoby projektowania i tworzenia witryn z uwzględnieniem standardów. Nauczysz się korzystać z języków XHTML, XML, CSS, ECMAScript oraz modelu DOM, które są wykorzystywane do pisania łatwych w pielęgnacji serwisów WWW. Przeczytasz również o mechanizmach ułatwień dostępu oraz mitach związanych z nimi. Jest to lektura obowiązkowa dla wszystkich projektantów i programistów, którzy chcą tworzyć nowoczesne witryny internetowe.

- Przegląd standardów sieciowych
- Projektowanie i tworzenie serwisów zgodnie ze standardami
- Walka z szybko starzejącymi się witrynami
- Nadawanie spójnego stylu witrynom za pomocą arkuszy CSS
- Pisanie przejrzystego kodu za pomocą języka XHTML
- Tworzenie skryptów manipulujących modelem DOM
- Obsługa różnych przeglądarek
- Stosowanie mechanizmów ułatwień dostępu

**Wykorzystaj sprawdzone techniki, które zaoszczędzą czas i pieniądze
zarówno Twoje, jak i użytkowników Twoich witryn**



Spis treści

Wprowadzenie	17
Nie wszystko dla wszystkich	17
Teoria a praktyka	19
Ciągłość, a nie zbiór sztywnych reguł	20
Pokazuj, nie sprzedawaj	20
Niech praca sprzedaje się sama	21
Sprzedaż wewnątrz firmy	22
Zapach zmian	23

Część I > Houston, mamy problem

Zanim zaczniesz	27
Nakręcanie kosztów, zmniejszanie zwrotów	28
Przerwanie cyklu starzenia się	30
Czym jest zgodność w przód?	31
Żadnych zasad, żadnego dogmatu	33
Praktyka, nie teoria	35
Czy ta podróż jest naprawdę potrzebna?	37
1 99,9% witryn wciąż jest przestarzałych	39
Nowoczesne przeglądarki i standardy sieciowe	40
Nowy kod do nowej pracy	42
Problem „wersji”	43
Myślenie wsteczne	46
Przestarzałe znaczniki: dodatkowy koszt dla właścicieli witryn	50
Zgodność wstecz	52
Blokowanie użytkowników nie wpływa dobrze na interesy	53
Droga do Paeanowa	57
Dobre traktowanie złego kodu	58
Lek	61

2	Projektowanie i budowanie z użyciem standardów	63
	Pokonywanie trudności	65
	Koszt projektowania przed wprowadzeniem standardów	66
	Nowoczesna strona starymi metodami	67
	Królestwo tragedii	71
	Trzy elementy standardów sieciowych	74
	Struktura	74
	Prezentacja	77
	Zachowanie	77
	W praktyce	78
	Zalety metod przejściowych	79
	Projekt standardów sieciowych: przenośność w zastosowaniu	81
	Jeden dokument dla wszystkich	82
	A List Apart: jedna strona, wiele widoków	84
	Projektowanie nie tylko z przeznaczeniem na ekran	86
	Oszczędność czasu i kosztów, wzrost zysków	87
	Co dalej?	88
	Przejściowa zgodność w przód (projektowanie hybrydowe)	88
	Całkowita zgodność w przód	91
3	Problem ze standardami	95
	Miło popatrzeć, trudno zakodować	95
	Wspólne zamiary, wspólne środki	97
	Przyjęcie standardów a rzeczywistość	98
	2000 — rok, w którym przeglądarki osiągnęły dojrzałość	100
	IE5/Mac: przełączanie i powiększanie	100
	Mocne posunięcie Netscape'a	101
	Przełamanie tamy	104
	Za mało, za późno?	105
	CSS: pierwsze koty za płoty	106
	Złe przeglądarki prowadzą do złych praktyk	107
	Kłątwa złego odwzorowywania	107
	Brak dziedziczenia	109
	Złe zachowanie	110
	Długo oczekiwany standard w językach skryptowych	111
	Mało czytelne witryny, niezrozumiałe nazewnictwo	112
	Problemy akademickie a problemy ekonomiczne	113
	Konsorcjum sugeruje, firmy sprzedają	114
	Świadomość produktu a świadomość standardów	114
	Słowo na F	116
	Wartość Flasha	117
	Problem z Flashem	119
	Inny problem z Flashem	119
	Zgodność to brzydkie słowo	120
	Potęga języka w formowaniu percepcji	120
	Problem z inspiracją	121
	Inne problemy	122

4 Wyszukiwanie, syndykacja, blogi, podkasty i długi ogon (oraz inne powody zwycięstwa standardów sieciowych)	125
Uniwersalny język (XML)	127
Porównanie XML-a i HTML-a	129
Jeden rodzic, wiele dzieci	129
Niezbędny element profesjonalnego oprogramowania	130
Bardziej popularny niż biały raper	131
Pięć spraw świadczących o potędze technologii	133
Złota żyła innowacji	134
Narzędzia do publikacji dla całej reszty	139
Do twoich usług	139
Aplikacje XML a twoja witryna	141
Kompatybilne z natury	142
Nowa era współpracy	143
Testy i specyfikacje	143
Jak można ze sobą współpracować?	144
Grupa robocza WHAT	145
Internet Explorer 7 i projekt standardów sieciowych	145
Standardy sieciowe i narzędzia edycyjne	146
Grupa zadaniowa Dreamweaver	146
Narzędzia WYSIWYG stają się pełnoletnie (dwa z trzech to nie najgorzej)	148
Od FrontPage do Expression Web Designer	148
Nadejście układów CSS	149
Kampania uaktualniania przeglądarek	150
Początek potopu	154
Skąd czerpać style?	155
Największa skarbnica wiedzy o CSS	158
Chwilowa moda... o ustalonym przeznaczeniu	158
Upowszechnianie standardów sieciowych	159
Witryny komercyjne dają się ponieść fali	162
Wired Digital zmienia technologię	162
Zachęcanie projektantów	164
Ciągłe pojawiają się nowe hity	164
Droga do sukcesu jest wybrukowana walidacją	165

Część II > Projektowanie i budowanie

5 Nowoczesny układ znaczników	171
Ukryty schemat kiepskiego kodu	176
Przeformułowanie czego?	178
Podsumowanie	180
Który XHTML jest dla mnie najlepszy?	180
XHTML 2 — nie dla każdego	181
10 najważniejszych powodów, dla których warto wybrać XHTML	182
5 powodów, dla których nie warto wybierać XHTML-a	183

6	XHTML: restrukturyzacja sieci	185
	Konwersja do XHTML-a: proste zasady, łatwe wytyczne	186
	Dokument rozpoczynaj od deklaracji DOCTYPE i przestrzeni nazw ..	186
	Zadeklaruj typ zawartości strony	189
	Wszystkie znaczniki pisz małymi literami	191
	Wartości wszystkich atrybutów umieszczaj w cudzysłowach	194
	Przypisuj wartości wszystkim atrybutom	195
	Zamykaj wszystkie znaczniki	196
	Zamykaj również „puste” znaczniki	196
	Nie umieszczaj podwójnych myślników w komentarzach	197
	Koduj wszystkie znaki < i &	198
	Podsumowanie zasad XHTML-a	198
	Kodowanie znaków: nudne, bardzo nudne i potwornie nudne	199
	Leczenie strukturalne	200
	Sensowne kodowanie dokumentu	201
	Elementy wizualne i struktura	205
7	Struktura w układzie ścisłym i hybrydowym: gwarancja zwartych i trwałych stron	207
	Czy każdy element musi być strukturalny?	208
	div, id i imi pomocnicy	209
	Semantyczny kod i wielokrotne użycie	214
	Układy hybrydowe i spójny kod: co należy, a czego nie wolno	218
	Nazwijmy złe rzeczy po imieniu	219
	Powszechne błędy w układach hybrydowych	219
	Znaczniki div są w porządku	223
	Pokoebać atrybut id	224
	Zakaz stosowania nadmiarowych komórek tabel	226
	Parada przestarzałych metod	227
	Czas map	227
	Niezadowolenie z map	228
	Brak dostępu, brak struktury	229
	Cięcie i składanie	229
	Dojrzewanie metody cięcia i składania	230
	Nadmierna rozwlekłość nadmiernie rozwlekłych tabel	232
	Powraca zły CSS	233
	Co dalej?	237
8	XHTML w przykładach: układ hybrydowy (część I)	239
	Zalety metod hybrydowych zastosowanych w tym rozdziale	239
	Arkusze stylów zamiast JavaScriptu	240
	Podstawowe podejście (wstęp)	241
	Oddzielne tabele: korzyści pod względem CSS i funkcji ułatwień dostępu	241
	Pomiń nawigację — co i jak	242
	Dodatkowe atrybuty id	247

Pierwszy kod taki sam jak ostatni	249
Kod nawigacji (pierwsza tabela)	249
Prezentacja, semantyka, czystość i grzech	250
Kod treści (druga tabela)	252
9 Podstawy CSS	253
Wstęp do CSS	253
Korzyści z CSS	254
Anatomia stylów	256
Selektory, deklaracje, właściwości i wartości	256
Wielokrotne deklaracje	257
Biała przestrzeń i brak rozpoznawania wielkości znaków	258
Wartości ogólne i alternatywne	259
Selektory grupowe	260
Dziedziczenie i jego przeciwnicy	260
Selektory potomne	262
Selektory id i selektory potomne	263
Selektory klas	264
Łączenie selektorów do tworzenia zaawansowanych efektów	265
Style zewnętrzne, osadzone i bezpośrednie	268
Zewnętrzne arkusze stylów	268
Style bezpośrednie	271
Metoda „najlepszego możliwego scenariusza”	272
Od stylów osadzonych do zewnętrznych: metoda dwóch arkuszy	272
Względne i absolutne ścieżki plików	275
Korzyści płynące ze stosowania metod najlepszego możliwego scenariusza i dwóch arkuszy stylów	275
10 Zastosowanie CSS: układ hybrydowy (część II)	277
Przygotowanie ilustracji	278
Ustalenie podstawowych parametrów	280
Style ogólne, więcej na temat skrótów i marginesów	280
Elementy niewidoczne i blokowe	281
Kolory odnośników (wprowadzamy pseudoklasy)	284
Poprawiamy inne pospolite elementy	286
Więcej na temat rozmiarów czcionek	288
Ustalenie układu strony	292
Elementy nawigacyjne: pierwsze podejście	295
CSS dla elementów nawigacyjnych: pierwsza próba przy drugim podejściu	299
CSS dla elementów nawigacyjnych: ostatnie podejście	301
Ostatnie kroki: style zewnętrzne oraz efekt „jesteś tutaj”	305

11 Praca z przeglądarką.	
Część I: przełączanie przez typ dokumentu i tryb standardowy	309
Saga o przełączaniu przez deklarację typu dokumentu	310
Sterowanie interpretacją w przeglądarce:	
przełącznik typu dokumentu	313
Pełna lista kompletnych deklaracji typu dokumentu XHTML	316
Świętujmy różnorodność przeglądarek!	
(A przynajmniej nauczymy się z nią żyć)	320
12 Praca z przeglądarką.	
Część II: model ramkowy, błędy i sposoby radzenia sobie z nimi ...	327
Model ramkowy i jego niedociągnięcia	328
Jak działa model ramkowy?	330
Jak model ramkowy został zepsuty?	331
Sztuczka z modelem ramkowym: CSS stanie się bardziej	
demokratyczny dzięki odpowiednim zabezpieczeniom	338
Błąd znaków odstępu w IE dla Windows	341
Błąd właściwości „float” w IE6 dla Windows	345
Float, Peek-a-Boo i co jeszcze	348
Flash i QuickTime: obiekty pożądanania?	349
Obiekty osadzone: opowieść o próżności i zemście	349
Dwie pieczenie na jednym ogniu: osadzanie obiektów	
multimedialnych przy przestrzeganiu standardów	351
Łyzka dziegieciu w beczce miodu: <object> nie działa	352
Świat, w którym omijanie błędów jest codziennością	353
13 Praca z przeglądarką. Część III: typografia	357
Rozmiar ma znaczenie	357
Kontrola użytkownika	358
Horrorzy starej szkoły	358
Punkty sporne	360
Nareszcie standardowy rozmiar	361
Wszelkie starania zniweczone przez jedno kliknięcie	364
Upojenie wężycieli: zła reakcja na zmiany w przeglądarkach	365
Standardowe rozmiary i najlepsze praktyki	367
Jednostki em: radość i płacz	368
Ustawienia użytkownika a jednostki em	368
Pasja do pikseli	369
Najmniejsza jednostka: to rzecz całkowicie względna	370
Kłopot z pikselami	372
Metoda symbolicznych wartości rozmiarów czcionek	373
Dlaczego wartości symboliczne wygrywiają z jednostkami em	
i procentami?	373
Początkowe problemy w implementacji wartości symbolicznych	374
Twój rozmiar	378

14 Podstawowe mechanizmy ułatwień dostępu	379
Dostępność według podręczników	381
Powszechna dezorientacja	383
Zły duch macza w tym palce	383
Prawo a elementy układu	387
Wyjaśniamy znaczenie paragrafu 508	388
Obalamy mity dostępności	390
Mit: dostępność zmusza cię do tworzenia dwóch wersji witryny	390
Mit: wersja tekstowa zaspokaja wymagania równego lub równorzędnego dostępu	391
Mit: dostępność kosztuje zbyt wiele	391
Mit: dostępność wymusza tworzenie prymitywnych, słabej jakości projektów	394
Mit: zgodnie z paragrafem 508 witryna musi wyglądać tak samo we wszystkich przeglądarkach i agentach użytkownika	394
Mit: dostępność jest „tylko dla osób niepełnosprawnych”	395
Mit: Dreamweaver MX/Cynthia Says/LIFT/ Tutaj wstaw nazwę narzędzia rozwiązuje wszystkie problemy dostępności	395
Mit: projektanci mogą swobodnie ignorować przepisy o dostępności, jeśli tak nakazują im klienci	396
Ułatwienia dostępu element po elemencie	397
Obrazki	397
Apple QuickTime i inne przesyłane strumieniowo obrazy wideo	400
Macromedia Flash 4/5	400
Macromedia Flash MX i Flash 8	401
Kolory	402
CSS	403
Efekty najezżdzenia oraz inne zachowania implementowane w skryptach	405
Formularze	407
Mapy obrazu	407
Układy oparte na tabelach	408
Tabele przechowujące dane	408
Ramki i aplety	408
Elementy błyskające lub migające	408
Sprawdzone narzędzia	409
Pokoehaj Cynthia	410
Zachowaj kolejność: nasz dobry znajomy atrybut tabindex	411
Planowanie dostępu: jak na tym skorzystasz	416
15 Wykorzystanie skryptów opartych na modelu DOM	419
DOM według podręczników	420
Co to jest DOM?	422
Standardowy sposób na to, by strony WWW zachowywały się jak aplikacje	422
Zatem gdzie to działa?	424

Proszę, DOM, nie zrób im krzywdy	425
Przełączniki stylów: ułatwiają dostęp, oferują wybór	429
16 Przeprojektowywanie z zastosowaniem CSS	433
Definiujemy wymagania	434
10 najważniejszych wymagań	434
Kładziemy fundamenty	436
Zabawa z nagłówkami	441
Zakończenie	444
> Dodatki	
<hr/>	
Biblioteczka standardów sieciowych	457
Skorowidz	449

99,9% witryn wciąż jest przestarzałych

Jest taka choroba, która dotyka niemal każdą witrynę znajdującą 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. Choć 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. 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 Firefox, Camino i Netscape Navigator) 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 pozostałe działają jedynie częściowo. 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, mam 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: zdobywca wielu nagród, darmowa przeglądarka Firefox 1.5+ oraz jej niemrawy komercyjny brat Netscape Navigator 8, Microsoft Internet Explorer 6 i nowsze dla Windows, Safari 2.0+ firmy Apple dla systemu Macintosh OS X oraz Opera 8.5+ firmy Opera Software. (Czy pominąłem Twoją ulubioną zgodną ze standardami przeglądarkę? Nie miałem zamiaru jej lekceważyć. Każda próba wyczenia wszystkich przeglądarek zgodnych ze standardami jest z góry skazana na niepowodzenie). Mimo że będę stosował 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.

Jednak brak perfekcji przeglądarek nie zwalnia z dążenia do zgodności ze standardami. Miliony ludzi nadal używają Internet Explorera 5 i 5.5 dla Windows. Jeśli chodzi o zachowanie standardów, te przeglądarki są gorsze od IE6+, Firefoxa, Opery i Safari. 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”.

Dziwny przypadek Internet Explorera 5 dla komputerów Macintosh

Przeglądarka Internet Explorer 5 w wersji dla komputerów Macintosh, w równym stopniu wielbiona co piętnowana przez projektantów, zajmuje szczególne miejsce w historii standardów sieciowych. W latach 1990-tych inżynier Tantek Çelik był reprezentantem Microsoftu w ramach grup roboczych W3C HTML i CSS. Kiedy jego przełożeni zleчили mu zarządzanie zespołem tworzącym Internet Explorera 5 dla komputerów Macintosh, Tantek włożył w to zdanie całą swoją wiedzę i pasję do standardów sieciowych. Współpracował nawet bezpośrednio z kluczowymi członkami projektu standardów sieciowych w celu przetestowania zgodności przeglądarki z nowatorskimi ówczesnie stronami opartymi na jednokolumnowym układzie CSS oraz obmyślenia rozszerzeń interfejsu użytkownika, dzięki którym strony byłyby bardziej przystępne. W efekcie powstał produkt, który obsługiwał standardy sieciowe bardziej dogłębnie i z większą dokładnością niż jakikolwiek inny przed nim. Internet Explorer 5 dla Macintoshy, cieszący się sporym uznaniem od momentu udostępnienia w 2000 roku, był pierwszą przeglądarką zgodną ze standardami.

Niefortunnie dla Tanteka zgodność ze standardami Internet Explorera 5 dla Macintoshy sprawiała, że Internet Explorer dla Windows wyglądał przy nim jak zacofany starszy brat. Microsoft wynagrodził Çelika, zabraniając mu dalszych prac nad przeglądarką, zamrażając w ten sposób wszystkie błędy. Błędy te wyszły na jaw dopiero wówczas, gdy na rynku pojawiły się inne przeglądarki obsługujące standardy sieciowe i projektanci zaczęli poddawać CSS coraz cięższym próbom. Jak można się spodziewać, skoro inne przeglądarki były rozwijane, a Internet Explorer 5 dla Macintosha nie, grupa jego użytkowników stopniała. W styczniu 2006 roku Microsoft oficjalnie pogrzebał tą przeglądarkę.

Współcześni projektanci zazwyczaj są bardziej sfrustrowani przez brak poprawnej obsługi złożonych układów CSS w tej przeglądarce niż zainspirowani przez jej pionierską przeszłość. Aby oddać sprawiedliwość tej przeglądarce, należy nadmienić że przestrzeganie standardów przez Internet Explorer 5 dla Macintoshy zmusiło Microsoft do znacznej poprawy obsługi standardów w wersji przeglądarki dla Windows. Gdyby tak się nie stało, nadal tworzylibyśmy układy bazujące na tabelkach.

Oprócz tego przydatne innowacje pochodzące z Internet Explorera 5 dla Macintoshy (jak narzędzie powiększenia tekstu) zostały zaadoptowane przez Firefoksa, Safari i większość innych nowoczesnych przeglądarek — aczkolwiek, co smutne, Microsoft nadal nie uznał za stosowne wprowadzenie ich do Internet Explorera dla Windows.

Podobnie 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 nie szczególne. (Wyjaśnimy to w drugiej części książki).

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 Firefox, Camino, Netscape 7+ i przeglądarki bazujące na silniku Gecko nie są nowszymi wersjami Netscape Navigatora 4. IE5+/Mac nie był uaktualnioną wersją IE4/Mac. Opera 7 nie bazuje na tym samym kodzie, który „napędzał” poprzednie wersje przeglądarek Opera. 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 zbyt 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. Był jakie wsparcie dla tak podstawowych standardów jak HTML stworzyło niejednorodne środowisko publikacji, a w konsekwencji niestrawne metody produkcji.

Kiedy pęka wyrostek robaczkowy u pacjenta, wykwalifikowani chirurdzy usuwają go całkowicie. Wyobraźmy sobie, co może się stać, kiedy żaden chirurg nie jest dostępny? Co będzie, kiedy pijany stażysta usunie zaledwie

połowę wyrostka, przy okazji dźgając kilka sąsiednich organów i na końcu zapominając zaszyć pacjenta? Porównanie jest trochę makabryczne, ale doskonale oddaje podejście do obsługi standardów w starych przeglądarkach: niebezpiecznie niekompetentne, 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 wersję tych reguł 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 Linux mieli pecha.

CSS był zaledwie jedną z wielu przyczyn problemów. 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 *produkcji* 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, których koszty nie są adekwatne do tego co oferują.

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.

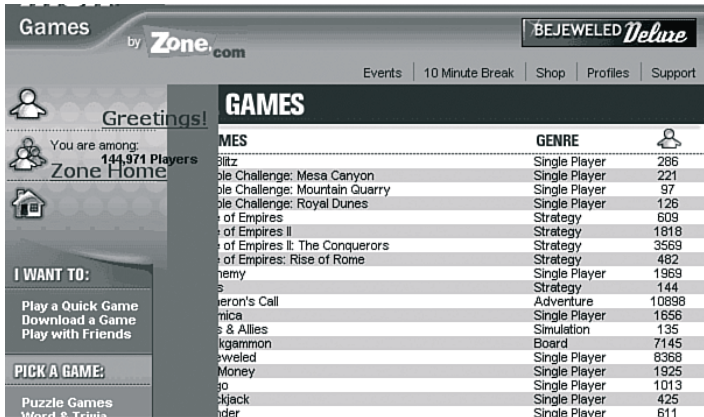
Mimo swojej kosztowności, bezsensowności i nietrwałości opisana praktyka nadal dominuje na rynku. Projektanci mający do czynienia z przeglądarką obsługującą standardy sieciowe traktują ją jak jedną z tych, które nie posiadają tej cechy. Tworzą kod, aby sprawdzić, czy jest to IE6, i „karmią” ją skryptami 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 nowych przeglądarek bazujących na silniku Mozilli, chociaż te również potrafią obsługiwać 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 systemów Windows i Macintosh 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 opeji 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 (przez co również jej zawartość jest trudniejsza do odnalezienia przez potencjalnych klientów). Stosowane strategie wywołują zatem często efekty, którym miały zapobiegać — niekonsekwentne prezentowanie witryn w różnych przeglądarkach (rysunki 1.1 i 1.2).

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

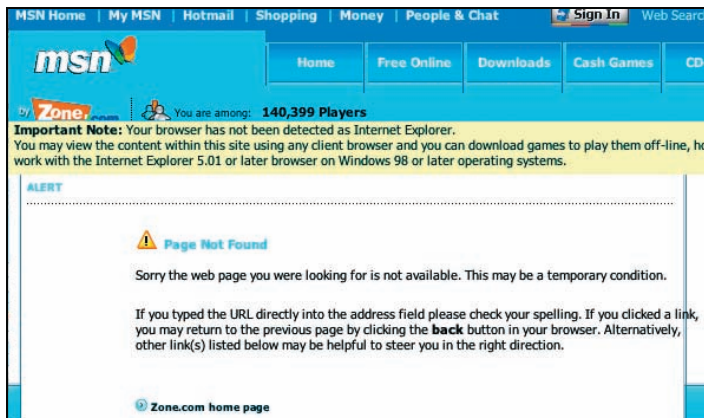


The screenshot shows the MSN Game Zone interface. At the top, it says "Games by Zone.com" and "BEJEWELLED Deluxe". There are navigation links for "Events", "10 Minute Break", "Shop", "Profiles", and "Support". A "Greetings!" section shows "You are among: 144,971 Players" and a "Zone Home" button. Below this is an "I WANT TO:" section with options: "Play a Quick Game", "Download a Game", and "Play with Friends". A "PICK A GAME:" section lists "Puzzle Games" and "Word & Trivia". The main content is a table of games:

TITLE	GENRE	PLAYERS
Blitz	Single Player	286
Blitz Challenge: Mesa Canyon	Single Player	221
Blitz Challenge: Mountain Quarry	Single Player	97
Blitz Challenge: Royal Dunes	Single Player	126
Age of Empires	Strategy	609
Age of Empires II	Strategy	1818
Age of Empires II: The Conquerors	Strategy	3569
Age of Empires: Rise of Rome	Strategy	482
Armed & Dangerous	Single Player	1969
Assassins	Strategy	144
Baron's Call	Adventure	10898
Bejeweled	Single Player	1856
Blitz & Allies	Simulation	135
Clash of Clans	Board	7145
Clash Royale	Single Player	8368
Clash Royale: Money	Single Player	1925
Clash Royale: Go	Single Player	1013
Clash Royale: Hack	Single Player	425
Clash Royale: Under	Single Player	611

Rysunek I.1.

W roku 2002 witryna MSN Game Zone (<http://zone.msn.com/>) obsługiwała 7 arkuszy stylów, prezentuje się jednak niepoprawnie w większości nowoczesnych przeglądarek. 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



The screenshot shows the MSN website with a navigation bar including "Home", "Free Online", "Downloads", "Cash Games", and "CD". Below the navigation bar, it says "by Zone.com" and "You are among: 140,399 Players". An "Important Note" states: "Your browser has not been detected as Internet Explorer. You may view the content within this site using any client browser and you can download games to play them off-line, however you must work with the Internet Explorer 5.01 or later browser on Windows 98 or later operating systems." Below this is an "ALERT" box with a warning icon and the text: "Page Not Found. Sorry the web page you were looking for is not available. This may be a temporary condition. If you typed the URL directly into the address field please check your spelling. If you clicked a link, you may return to the previous page by clicking the back button in your browser. Alternatively, other link(s) listed below may be helpful to steer you in the right direction." At the bottom of the alert box is a link: "Zone.com home page".

Rysunek I.2.

Uczciwie przyznaję, że poprzedni zrzut z ekranu pochodzi sprzed 4 lat. Obecnie ta sama strona wygląda jeszcze gorzej, gdy wrzucono na nią jeszcze więcej kodu. Sześć lat po tym, kiedy Microsoft wypuścił na rynek pierwszą przeglądarkę zgodną ze standardami, niektóre fragmenty witryny Microsoftu nadal nie przestrzegają podstawowych zasad projektowania w zgodzie ze standardami sieciowymi

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 „bezprowadową” wersję swojej strony tylko po to, aby przekonać się, że zastosowany przez nich język znaczników jest przestarzały lub strona nie funkcjonuje w nowym urządzeniu 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 jakiegokolwiek większej komercyjnej strony, takiej jak Amazon.com lub eBay.com. Zbadaj ich zawily 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 jakiegokolwiek 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.

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”).

Znam wiele firm, które wydały ponad milion dolarów na przesadnie złożone, ale niespecjalnie użyteczne systemy zarządzania treścią (CMS, ang. *content management system*). Twórcy takich monstrualnych programów częściowo usprawiedliwiają ich oburzające koszty, wskazując na możliwość mozolnego generowania wszelkiego rodzaju niestandardowych wersji kodu. Poza marnotrawstwem ogromnych ilości gotówki taka praktyka szkodzi także możliwości przeszukiwania takich stron przez zatopienie sensownych treści w morzu nieznaczających znaczników (rysunek 1.3). Najbardziej ze wszystkich taka praktyka wystawia na próbę cierpliwość użytkownika korzystającego z modemowego dostępu do sieci przez zapychanie łącza rozwidlonym kodem, zagnieżdżonymi tabelami, przeróżnymi sztuczkami z obrazkami, a także przestarzałymi znacznikami i atrybutami.

```
<map name=m><area alt="My Yahoo!" coords="44,0,106,47" href=r/il><area alt=Finance coords="121,0,170,47" href=r/fl><area alt=Shop coords="192,0,241,47" href=r/xm><area alt=Email coords="493,0,542,47" href=r/ml><area alt=Messenger coords="558,0,618,47" href=r/pl><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>&nbsp;</td><td bgcolor=cccccc width=1><spacer type=block width=1 height=16></td><td align=center width=20>&nbsp;</td><td width=18></td><td nowrap><font size=-1 face=arial>&nbsp;<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><table cellpadding=0 cellspacing=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>&nbsp;<input type=submit value=Search>&nbsp;&nbsp;</td><td><font face=arial size=-2>&#149; <a href=r/so>advanced search</a><br>&#149; <a href=r/zl>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% cellpadding=0 border=0><tr><td colspan=3><table width=100% cellpadding=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><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><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>
```

Rysunek 1.3.

Szybko! Spróbuj znaleźć na tej stronie ważne dla Yahoo informacje o „Osobistości roku” wśród jej zagmatwanej, niestrukturalnej składni HTML z 2001 roku. Podpowieź: Jeśli ty masz trudności z jej odnalezieniem, tak samo trudne będzie to dla czytelników i wyszukiwarek (Yahoo.com)

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” powyżej).

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 unikać.

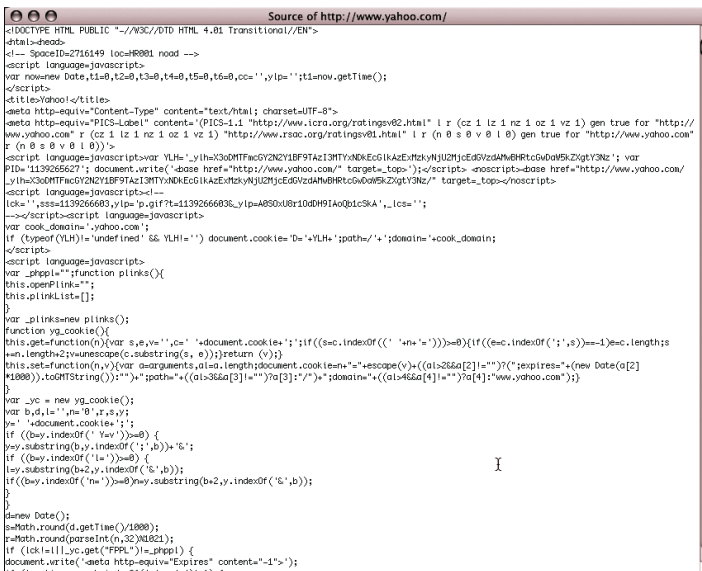
Liczby nie kłamią. Jeżeli strona zredukuje swój kod o 35%, zredukuje również o tyle samo koszty utrzymania łącza. Firma wydająca 10 000 zł rocznie mogłaby zaoszczędzić 3 500 zł. Przy wydatkach rzędu 640 000 zł oszczędności wynoszą 224 000 zł.

Strona domowa Yahoo (rysunek 1.4) 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 `` (rysunki 1.3 i 1.5) przez wydajne style CSS, koszt ładowania każdej strony zmalałby wielokrotnie, a zyski firmy w konsekwencji wzrosłyby. Dlaczego zatem Yahoo nie wykonało takiego kroku?

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 weale szacie graficznej, lecz oferowanym usługom.



Rysunek I.4.
Strona domowa
Yahoo (<http://www.yahoo.com/>)



Rysunek I.5.
Yahoo od środka.
Zobacz źródło,
a przekonasz się,
że kod służący
do stworzenia
tej prosto
wyglądającej strony
jest niewyobraźalnie
skomplikowany.
Mimo że Yahoo
zaczęło w końcu
używać CSS, robi
to w najmniej wydajny
sposób — nadal używa
znaczników ``
oprócz reguł CSS!

Przykład tej, skądinąd interesującej, firmy (marnującej swoje łącza na dostarczanie wyglądu i zachowania¹, 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.

Przestarzałe znaczniki: dodatkowy koszt dla właścicieli witryn

Załóżmy, że kod jednej strony zbudowanej według starych zasad zajmuje 60 kB. Zastąpienie znaczników `` 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 (w tym większość blogów).

¹ Wszystkich interaktywnych cech witryny, jakie można stworzyć przy użyciu HTML-a i JavaScriptu — *przypp. thum*.

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. Obciążenie rozmiaru plików o połowę sprowadza wysokość płaconych rachunków do przyzwoitego poziomu.

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 plik 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.

Jak to działa: Wiele firm oferujących usługi hostingowe przydziela swoim użytkownikom „wolny” od opłat miesięczny limit transferu plików — na przykład do 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 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. 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 4,95 dolarów za każdy megabajt ponad limit. Większe witryny, z większym ruchem na stronach oszczędzają najwięcej przez redukcję 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 odpowiadają: „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 Firefox 8.5. Zasada „zgodności wstecz” — traktowana jako święty Graal programowania — brzmi niezłe w teorii. Jednak jej koszt jest zbyt wysoki, a ona sama od zawsze opiera się na fałszywym założeniu.

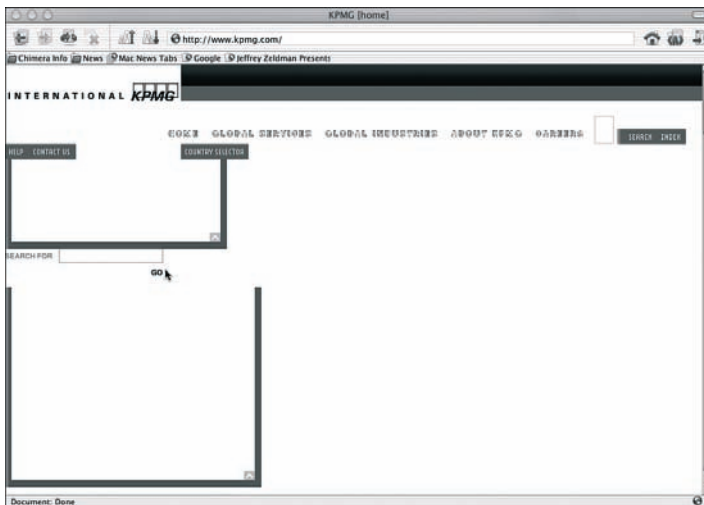
Nie istnieje prawdziwa zgodność wstecz. Zawsze istnieje punkt „odejścia”. Na przykład ani Mosaic (pierwsza przeglądarka graficzna), ani Netscape 1.0 nie obsługują układów opartych na tabelach HTML-owych. Zatem użytkownicy tych archaicznych przeglądarek nie mogą zobaczyć tego samego, co użytkownicy odrobinę 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

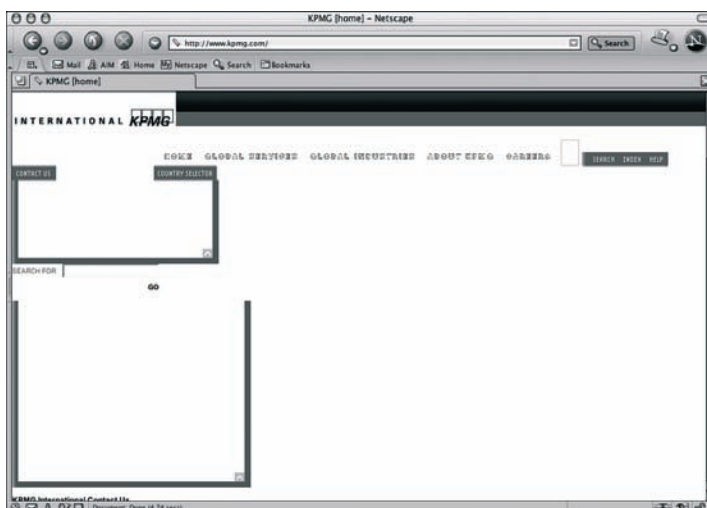
Podczas gdy niektóre firmy dokonują zamachu na swoje dochody, próbując zapewnić wsparcie nawet dla najstarszej przeglądarki, inne decydują się na obsługę wyłącznie jednej z nich. Ze względu na błędne założenia rośnie liczba stron projektowanych wyłącznie do współpracy z Internet Explorerem (czasem wyłącznie na platformie Windows), blokując tym samym 15 – 25% potencjalnych użytkowników i klientów (rysunki 1.6, 1.7, 1.8, 1.9, 1.10 i 1.11).



Rysunek 1.6. Strona domowa KPMG (<http://www.kpmg.com/>) z roku 2003 przeglądana w Navigatorze. Zupełna rozsyпка układu jest efektem zastosowania kodu działającego wyłącznie w IE

Nie będę udawać, że rozumiem podejście biznesowe firmy, która z założenia mówi NIE jednej czwartej swoich potencjalnych klientów. Tak duża liczba klientów stracona przez krótkowzroczne podejście nie powinna być

Rysunek I.7.
Serwis KPMG jest
równie beużyteczny
w Netscapie 7

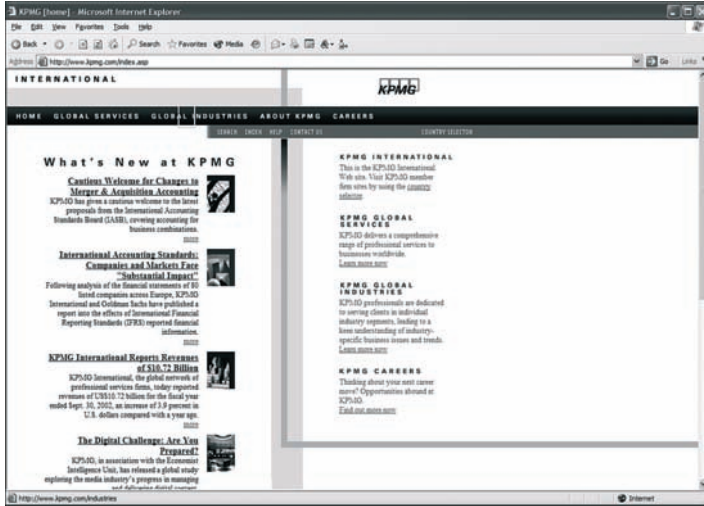


Rysunek I.8.
Jeśli ta strona była
przeznaczona tylko
dla Internet Explorera,
to jak działała
w Internet Explorerze 5
dla komputerów
Macintosh? Jak widać,
nie za bardzo



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 gwarancja, że Internet Explorer (lub nawet przeglądarki dla komputerów stacjonarnych jako kategoria oprogramowania)

**Rysunek I.9.**

Ta sama strona widziana przez IE6/Windows. Tutaj serwis w końcu działa!

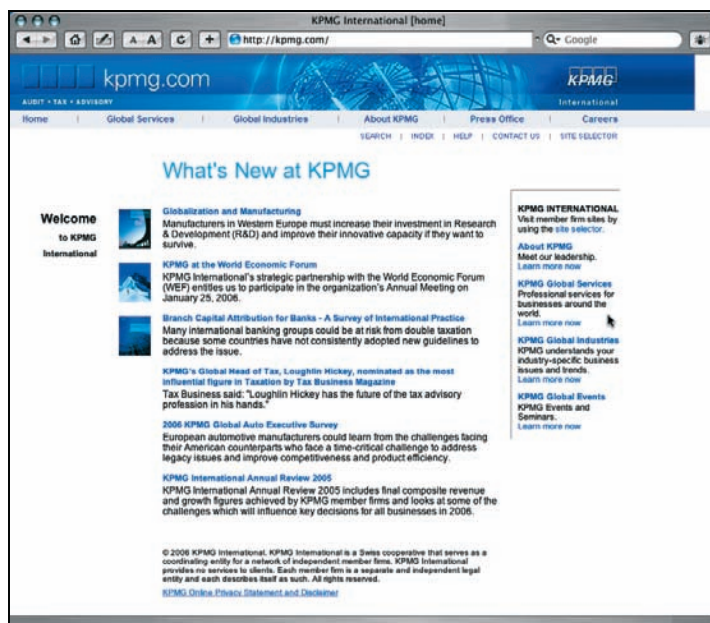
**Rysunek I.10.**

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

będą dominować w przyszłości. Z jednej strony, w chwili pisania tej książki Firefox nieustannie odbiera udziały w ranku Internet Explorer. Z drugiej — coraz więcej osób korzysta z internetu przy użyciu urządzeń mobilnych. W Stanach Zjednoczonych liczba urządzeń stacjonarnych podłączonych do internetu nadal przewyższa liczbę urządzeń mobilnych — w Japonii sytuacja jest odwrotna. Mimo że stosunek ten ciągle się zmienia, ogólny trend wskazuje na urządzenia mobilne (www.gotmobile.com). W miarę jak wszechobecny dostęp do internetu zyskuje akceptację i stwarza nowe rynki zbytu, pomysł uwzględniania w projekcie dziać *jakiegokolwiek* przeglądarki wydaje się coraz bardziej przeżytkiem 20-tego wieku i staje się coraz mniej sensowny.

Rysunek I.11.

Po gruntownym przeprojektowaniu nowa strona KPMG wygląda poprawnie i działa dobrze w wielu przeglądarkach na różnych platformach. Stało się tak za sprawą uaktualnionej składni, nieograniczonej tylko do Internet Explorera



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, waleczący 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 zadawały „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 Netscapie 4, nie działało w Netscapie 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, niezgadający się z tą sytuacją, nie mieli alternatywy, musieli zagryźć zęby i stworzyć 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 4.0, 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 bardzo gorącym towarem, a klienci chętnie płacili bardzo wysokie rachunki, dlatego 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 myślały 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 (obecnie Adobe). Jak na biznes kreowany przez wizję przemysł sieciowy potrafi być nadzwyczajnie krótkowzroczny.